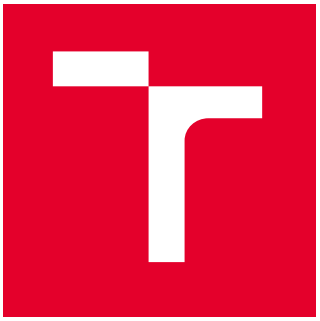


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ

FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY

INSTITUTE OF INFORMATICS

NÁVRH WEBOVÉ APLIKACE PRO START-UP

WEB APPLICATION DEVELOPMENT FOR A START-UP BUSINESS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Dominik Deminger**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jan Luhan, Ph.D., MSc**

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání bakalářské práce

Ústav:	Ústav informatiky
Student:	**Dominik Deminger**
Vedoucí práce:	**Ing. Jan Luhan, Ph.D., MSc**
Akademický rok:	2023/24
Studijní program:	Manažerská informatika

Garant studijního programu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává bakalářskou práci s názvem:

Návrh webové aplikace pro start–up

Charakteristika problematiky úkolu:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Vlastní návrhy řešení
Závěr
Seznam použité literatury
Přílohy

Cíle, kterých má být dosaženo:

Návrh webové aplikace pro konkrétní start–up subjekt a vytvoření jejího funkčního prototypu na základě konkrétních požadavků zadavatele.

Základní literární prameny:

BRUCKNER, T., J. VOŘÍŠEK, A. BUCHALCEVOVÁ a kol. Tvorba informačních systémů: Principy, metodiky, architektury. Praha: Grada Publishing, 2012. 360 s. ISBN 978-80-247-4153-6.

LAUDON, K. C. and C. G. TRAVER. E-commerce 2017: Business, Technology, Society. 13th ed. Boston: Pearson, 2017. 912 p. ISBN 978-0-13-460156-4.

SCHWALBE, K. Řízení projektů v IT: Kompletní průvodce. Praha: Computer Press, 2011. 632 s. ISBN 978-80-251-2882-4.

WELLING, L. a L. THOMSON. Mistrovství PHP a MySQL. 1. vyd. Brno: Computer Press, 2017. 799 s. ISBN 978-80-251-4892-1.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2023/24

V Brně dne 4.2.2024

L. S.

Ing. Jiří Kříž, Ph.D. garant	doc. Ing. Vojtěch Bartoš, Ph.D. děkan

Abstrakt

Bakalářská práce se zabývá návrhem prototypu webové aplikace za pomoci Node.js a dalších technologií a postupů používaných pro vývoj a implementaci. Hlavním cílem práce je vytvoření funkčního prototypu internetového obchodu pro společnost Joynda s.r.o., který bude splňovat požadavky majitele. V textu jsou podrobně popsány technologie a postupy použité k projektu, zdrojový kód, implementace a detailně popsána struktura stránek a jejich funkcionalita.

Klíčová slova

sql, node.js, react, typescript, mysql, databáze, webová aplikace, uml, erd, mvc, backend, frontend

Abstract

The bachelor thesis deals with the design of a prototype web application using Node.js and other technologies and procedures commonly used for development and implementation. The main goal of the thesis is to create a functional prototype of an online store for the company Joynda s.r.o., which will meet the owner's requirements. The text provides detailed descriptions of the technologies and procedures used in the project, the source code, implementation and a detailed description of the structure of the pages and their functionality.

Keywords

sql, node.js, react, typescript, mysql, databases, web application, uml, erd, mvc, backend, frontend

Bibliografická citace

DEMINGER, Dominik. *Návrh webové aplikace pro start-up* [online]. Brno, 2024 [cit. 2024-05-10]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/160481. Bakalářská práce. Vysoké učení technické v Brně, Fakulta podnikatelská, Ústav informatiky. Vedoucí práce Ing. Jan Luhan, Ph.D., MSc.

Čestné prohlášení

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 10. 5. 2024

Dominik Deminger

autor

Obsah

Úvod

Žijeme v době technologií, ve které si málokdo dokáže představit každodenní život bez vynálezů moderní doby. Technologie se staly našimi věrnými pomocníky jak v osobním životě, tak ve firemním prostředí. Firmy se velmi dobře adaptovaly dnešní době a pro svůj chod využívají širokou škálu nástrojů, které jim nejen usnadňují chod firmy, ale i zvyšují její zisky. Díky webovým aplikacím firmy mohou poskytovat své služby celosvětově, bez nutnosti fyzických prodejen, a na různých platformách, jako jsou telefony, počítače nebo tablety.

Předmětem této bakalářské práce bylo vytvořit webovou aplikaci (internetový obchod) pro firmu Joynda s.r.o., která se zabývá především prodejem oblečení a módních doplňků. Jedná se o začínající firmu, která neočekává příliš velkou zátěž na stránku. Společnost v tuto chvíli nemá v plánu masovou výrobu ani prodej. Naopak si chce spíše vytvořit menší, ale zato loajální klientelu, která se bude k firmě vracet. Podle tohoto byla aplikace navrhovaná a implementovaná, aby splňovala všechny požadavky, a mohla co nejdříve do produkce. Hlavním cílem bylo vytvořit aplikaci, která co nejvíce zpříjemní zákazníkovi jeho požitek z nákupu, a umožní mu co nejjednodušší nákupní proces.

Cíle práce, metody a postupy zpracování

Cílem této práce bylo navrhnout a implementovat funkční prototyp webové aplikace zaměřené na prodej oblečení a módních doplňků (internetový obchod). Jedná se o aplikaci vyvíjenou pro společnost Joynda s.r.o., která se specializuje na prodej módních produktů. Společnost je malá a začínající, aktuálně nabízí omezený sortiment produktů, ale plánuje expandovat. S tímto faktem jsem při návrhu aplikace počítal, aby mohla být následně snadno rozšířena a upravena podle potřeb firmy.

Při vypracování práce byla nejprve zpracována teoretická část z odborné literatury, která posloužila k analýze technologií a postupů vhodných pro tvorbu aplikace. Pro účely aplikace byla zvolena platforma Node.js, která zajišťuje obsluhu požadavků uživatele doplněná o MySQL databázi pro ukládání potřebných dat a knihovna React, která slouží k vizualizaci těchto dat pro uživatele. Dalším krokem bylo získání a sjednocení myšlenek majitele, podle kterých byly definovány požadavky na aplikaci a její funkcionalitu. Pomocí diagramu užití byly následně detailně popsány jednotlivé funkce a rozhraní aplikace podle majitelových požadavků.

1 Teoretická východiska práce

V následující části mé práce popisuji základní pojmy týkající se tvorby a implementace webové aplikace, stejně jako technologie a metody používané v této oblasti. Výsledkem práce má být funkční prototyp aplikace, který splňuje všechny potřeby zadavatele ohledně funkcionality a požadovaných rozhraní. Tento prototyp by měla firma po případných upravách mít možnost použít pro své podnikání.

1.1 Webová aplikace

"Webová aplikace (web app) je program, který je uložen na vzdáleném serveru a dodáván přes internet prostřednictvím rozhraní prohlížeče. Webové služby jsou definovány jako webové aplikace a mnoho, i když ne všechny, webové stránky obsahují webové aplikace.

Vývojáři navrhují webové aplikace pro širokou škálu použití a uživatelů, od organizace až po jednotlivce z mnoha důvodů. Běžně používané webové aplikace mohou zahrnovat webové poštovní schránky, online kalkulačky nebo obchody s elektronickým obchodováním. Zatímco některé webové aplikace mohou být přístupné pouze pomocí určitého prohlížeče, většina je dostupná bez ohledu na prohlížeč.

Jak fungují webové aplikace

Webové aplikace nemusí být stahovány, protože jsou přístupné přes síť. Uživatelé mohou přistupovat k webové aplikaci prostřednictvím webového prohlížeče, jako je Google Chrome, Mozilla Firefox nebo Safari.

Pro fungování webové aplikace potřebuje webový server, aplikační server a databázi. Webové servery spravují požadavky přicházející od klienta, zatímco aplikační server dokončuje požadovanou úlohu. Databáze ukládá veškeré potřebné informace.

Webové aplikace obvykle mají krátké vývojové cykly a malé vývojové týmy. Většina webových aplikací je psána v JavaScriptu, HTML5 nebo CSS. Klientské programování obvykle využívá tyto jazyky, které pomáhají vytvářet uživatelské rozhraní aplikace. Serverové programování vytváří skripty, které webová aplikace používá." (1)

1.2 Databáze

„Od svého vzniku patří databáze mezi nejvíce zkoumané oblasti znalostí v počítačové vědě. Databáze je úložiště dat, navržené k podpoře efektivního ukládání, získávání a údržby dat. Existuje několik typů databází, které odpovídají různým požadavkům průmyslu. Databáze může být specializovaná na ukládání binárních souborů, dokumentů, obrázků, videí, relačních dat, multidimenzionálních dat, transakčních dat, analytických dat nebo geografických dat, abychom jmenovali jen několik.

Data mohou být ukládána ve různých formách, konkrétně ve formách tabulek, hierarchií a grafů. Pokud jsou data uložena ve formě tabulek, nazývá se to relační databáze. Když jsou data organizována ve formě stromové struktury, nazývá se to hierarchická databáze. Data uložená jako grafy, které reprezentují vztahy mezi objekty, se nazývají síťová databáze." (2, s. 23)

V dnešní době téměř každá aplikace využívá nějaký typ databáze pro své fungování. Databáze se mohou lišit v způsobu manipulace s daty i ve způsobu skladování dat. Dobrou ukázkou je rozdíl mezi relační a nerelační databází. Relační databáze ukládají svá data ve formě tabulek složených z řádků a sloupců, uchovávají vztahy mezi tabulkami pomocí klíčů, které tyto vztahy definují. Na druhou stranu nerelační databáze ukládají svá data převážně v souborech, které nejsou nijak propojeny a nemají mezi sebou pevně definované vztahy. (2; 3)

Existuje velké množství databází ze kterých je možné volit, jako jsou například PostgreSQL, MongoDB nebo Oracle. Pro potřeby projektu jsem zvolil relační databázi MySQL. MySQL databáze byla zvolena díky typu dat, které jsou do aplikace ukládány. Jedná se o data jako jsou informace o produktech, nebo údaje o uživatelských účtech,

které mají pevně dané vazby mezi sebou, a díky relační databázy je můžeme snadno definovat. (2; 3; 4)

1.2.1 MySQL

"MySQL je nejoblíbenější open source databázový systém na světě. Podle DB-Engines se MySQL řadí na druhé místo mezi nejpoužívanějšími databázemi, hned za Oracle Database. MySQL pohání mnoho z nejvíce navštěvovaných aplikací, včetně Facebooku, Twitteru, Netflixu, Uberu, Airbnb, Shopify a Booking.com.

Díky tomu, že je MySQL open source, zahrnuje mnoho funkcí vyvinutých ve spolupráci s uživateli po více než 25 let. Je tedy velmi pravděpodobné, že vámi oblíbená aplikace nebo programovací jazyk je podporován MySQL databází. (...) Databáze jsou základním úložištěm dat pro všechny softwarové aplikace. Například když někdo provádí webové vyhledávání, přihlašuje se do účtu nebo dokončuje transakci, databázový systém ukládá informace tak, aby byly přístupné v budoucnosti.

Relační databáze ukládá data do oddělených tabulek místo toho, aby všechna data byla umístěna v jednom velkém úložišti. Struktura databáze je organizována do fyzických souborů optimalizovaných pro rychlost. Logický datový model s objekty jako jsou datové tabulky, pohledy, řádky a sloupce nabízí flexibilní programovací prostředí. Nastavujete pravidla upravující vztahy mezi různými datovými poli, jako je jedna k jedné, jedna k mnoha, unikátní, povinná nebo nepovinná a "ukazatele" mezi různými tabulkami. Databáze dodržuje tato pravidla tak, že s dobře navrženou databází vaše aplikace nikdy nevidí data, která jsou nekonzistentní, zdvojená, opuštěná, zastaralá nebo chybějící.

Část "SQL" v "MySQL" znamená "Structured Query Language" neboli "Strukturovaný dotazovací jazyk". SQL je nejběžnějším standardizovaným jazykem používaným k přístupu k databázím. V závislosti na vašem programovacím prostředí byste mohli zadávat SQL přímo (například pro generování zpráv), vkládat SQL příkazy do kódu napsaného v jiném jazyce nebo používat jazykem-specifické API, které skrývá syntaxi SQL." (5)

SQL je nástroj primárně používaný pro získávání dat z databáze pomocí dotazů. Toto ale není jediná funkce SQL. SQL také umožňuje vytváření databází, vkládání dat do tabulek, mazání dat a mnohem více. Jednou ze zajímavějších funkcí SQL jsou tzv. spouštěče (triggery), což jsou předdefinované funkce, které se spouštějí před nebo po provedení akce nad databází. Na následujícím obrázku je ukázka jednoduchého dotazu a následného výsledku. Databáze má za úkol vybrat všechna data z tabulky 'product', která mají aktivní status. (5; 6)

```
SELECT * FROM products WHERE products.status = "Active";
```

Obr. 1 Ukázka SQL dotazu

Zdroj: autor bakalářské práce

	id	collection_id	name	price	discount	description	add_date	last_update	status
Edit Copy Delete	1161	273	triko1	10	0	Lorem ipsum	2024-02-26 16:21:52	2024-02-29 22:16:43	Active
Edit Copy Delete	1162	272	triko2	10	0	Lorem ipsum	2024-02-28 00:26:15	2024-02-28 00:26:15	Active
Edit Copy Delete	1163	NULL	triko3	25	0	Lorem ipsum	2024-02-28 00:26:41	2024-02-28 00:26:41	Active
Edit Copy Delete	1164	NULL	triko4	25	0	Lorem ipsum	2024-02-28 00:33:43	2024-02-28 00:37:01	Active
Edit Copy Delete	1165	272	triko5	15	0	Lorem ipsum	2024-02-28 18:11:11	2024-02-28 18:44:43	Active
Edit Copy Delete	1171	NULL	triko6	20	0	Lorem ipsum	2024-03-01 15:18:36	2024-03-01 15:18:36	Active
Edit Copy Delete	1172	272	triko7	15	0	Lorem ipsum	2024-03-01 15:58:30	2024-03-01 15:58:30	Active
Edit Copy Delete	1173	272	triko8	40	0	Lorem ipsum	2024-03-03 22:06:16	2024-03-03 22:06:16	Active

Obr. 2 Výsledek SQL dotazu

Zdroj: Aplikace phpMyAdmin

1.3 Použité technologie

Aplikace je celá postavena na programovacím jazyce JavaScript, který je převážně používán pro vývoj webových aplikací. V následující části popisuji jak samotný jazyk, tak ale i jeho rozšíření, které mu umožňují provádět úkony přesahující jeho původní zaměry.

1.3.1 Javascript

„JavaScript byl představen v roce 1995 jako způsob, jak přidávat programy do webových stránek v prohlížeči Netscape Navigator. Jazyk od té doby přijaly všechny ostatní hlavní grafické webové prohlížeče. Umožnil vznik moderních webových aplikací - tedy aplikací, se kterými můžete interagovat přímo, aniž by se pro každou akci musela znovu načíst stránka. JavaScript se také používá v tradičnějších webových stránkách k poskytování různých forem interaktivity a chytrostí.

Je důležité si uvědomit, že JavaScript má téměř nic společného s programovacím jazykem jménem Java. Podobné jméno bylo inspirováno marketingovými úvahami spíše než rozumným úsudkem. Když se představoval JavaScript, programovací jazyk Java byl intenzivně propagován a získával popularitu. Někdo si myslel, že je dobrý nápad se na tomto úspěchu spoléhat. Teď jsme ale uvězněni s tímto jménem.

Po jeho přijetí mimo Netscape byl napsán standardní dokument, který popisuje způsob, jak by měl jazyk JavaScript fungovat, aby se různé části softwaru tvrdící, že podporují JavaScript, mohly ujistit, že skutečně poskytují stejný jazyk. Tento dokument se nazývá standard ECMAScript, podle organizace Ecma International, která provedla standardizaci. V praxi lze termíny ECMAScript a JavaScript používat zaměnitelně – jsou to dva názvy pro stejný jazyk." (7)

1.3.2 Typescript

„Více než 20 let po svém uvedení do programátorské komunity je JavaScript nyní jedním z nejrozšířenějších jazyků pro více platforem, které kdy byly vytvořeny. Začínaje jako malý skriptovací jazyk pro přidávání triviální interaktivity na webové stránky, JavaScript vyrostl do jazyka volby pro aplikace na frontendu i backendu všech velikostí. Zatímco velikost, rozsah a složitost programů psaných v JavaScriptu vzrostla exponenciálně, schopnost jazyka JavaScript vyjadřovat vztahy mezi různými jednotkami kódu se nezvýšila. Spolu s poměrně zvláštními semantikami běhu JavaScriptu tato neshoda mezi jazykem a složitostí programu způsobila, že vývoj v JavaScriptu je obtížný úkol při správě větších projektů.

Nejběžnější druhy chyb, které programátoři píší, lze popsat jako chyby typů: byl použit určitý typ hodnoty, kde byl očekáván jiný typ hodnoty. To může být způsobeno jednoduchými překlepy, selháním porozumění ploše API knihovny, nesprávnými předpoklady o běhu programu nebo jinými chybami. Cílem TypeScriptu je být statickým typovým kontrolérem pro programy v JavaScriptu - jinými slovy, nástrojem, který běží před spuštěním vašeho kódu (statický) a zajišťuje, že typy programu jsou správné (typechecked).“ (8)

1.3.3 React

“ReactJS je open-source, komponentová knihovna pro frontend, která je zodpovědná pouze za vrstvu zobrazení aplikace. Je udržována společností Facebook.

ReactJS používá mechanismus založený na virtuálním DOM k vyplnění dat (zobrazení) do HTML DOM. Virtuální DOM pracuje rychle díky skutečnosti, že mění pouze jednotlivé prvky DOM místo toho, aby každým spuštěním znovu načítal celý DOM.

Aplikace React se skládá z několika komponent, z nichž každá je zodpovědná za výstup malého, opakovaně použitelného kusu HTML. Komponenty mohou být vnořeny do jiných komponent, aby umožnily sestavení složitých aplikací z jednoduchých stavebních bloků. Komponenta může také udržovat vnitřní stav - například komponenta TabList může uchovávat proměnnou odpovídající právě otevřené záložce.

React nám umožňuje psát komponenty pomocí doménově specifického jazyka nazvaného JSX. JSX nám umožňuje psát naše komponenty pomocí HTML a zároveň do nich mísit události JavaScriptu. React to interně převede na virtuální DOM a nakonec pro nás vytvoří HTML.

React "reaguje" na změny stavu ve vašich komponentách rychle a automaticky tak, že znovu vykreslí komponenty v HTML DOM pomocí virtuálního DOM. Virtuální DOM je reprezentace skutečného DOM v paměti. Tím, že většinu zpracování provádí uvnitř virtuálního DOM namísto přímo v DOM prohlížeče, může React rychle jednat a přidávat,

aktualizovat a odebírat komponenty, které se od posledního cyklu vykreslování změnily." (9, s. 2)

1.3.4 Node.js

"Node vznikl, když původní vývojáři vzali JavaScript, což byl obvykle jazyk spouštěný pouze v prohlížeči, a umožnili mu běžet na vašem počítači jako samostatný proces. To znamená, že jsme mohli vytvářet aplikace pomocí JavaScriptu mimo kontext prohlížeče. Dříve měl JavaScript omezenou sadu funkcí. Když jsem ho používal v prohlížeči, mohl jsem dělat věci jako aktualizovat URL a odstranit loga Node, přidávat události kliknutí nebo cokoli jiného, ale opravdu jsem nemohl dělat mnohem více. S Node nyní máme sadu funkcí, která vypadá mnohem podobněji jako u jiných jazyků, jako je Java, Python nebo PHP. Některé z těchto funkcí jsou následující:

- *Můžeme psát aplikace Node pomocí syntaxe JavaScriptu*
- *Můžete manipulovat se svým souborovým systémem, vytvářet a odstraňovat složky*
- *Můžete přímo vytvářet dotazování databází*
- *Dokonce můžete vytvářet webové servery pomocí Node*

To byly věci, které v minulosti nebyly možné, a to díky Node. Nyní jak Node, tak JavaScript, který se provádí uvnitř vašeho prohlížeče, běží na stejném stroji. Nazývá se to stroj V8 pro spouštění JavaScriptu. Je to otevřený zdrojový stroj, který vezme JavaScriptový kód a přeloží ho do mnohem rychlejšího strojového kódu. A to je velká část toho, co dělá Node.js tak rychlým. Strojový kód je nízkoúrovňový kód, který váš počítač může spouštět přímo, aniž by ho musel interpretovat. Váš počítač zná pouze určité typy kódu, například váš počítač nemůže spouštět JavaScriptový kód nebo PHP kód přímo, aniž by ho nejprve převedl do nízkoúrovňového kódu. Pomocí tohoto stroje V8 můžeme vzít náš JavaScriptový kód, přeložit ho do mnohem rychlejšího strojového kódu a spustit ho. A to je místo, kde vstupují všechny ty nové funkce. Stroj V8 je napsán v jazyce

nazývaném C++. Takže pokud chcete rozšířit jazyk Node, nepíšete kód Node, ale píšete kód C++, který staví na tom, co již V8 má.“ (10, s. 15)

1.4 Použité aplikace třetích stran

Samotná aplikace je vyvíjena na počítači programátora. Zde se implementuje samotná funkcionalita aplikace a provádí se testování. Aby naši aplikaci mohl navštívit uživatel, musíme využít některé z nabízených možností, které umožní přístup k aplikaci prostřednictvím internetového prohlížeče.

1.4.1 GitHub

GitHub je platforma pro správu verzí a hostování softwarových projektů. GitHub umožňuje vytvářet repozitáře (složky), do kterých je možné nahrát jakékoliv části aplikace. Podporuje Git, což je systém pro kontrolu změn v repozitářích. Git umožňuje nejen sledovat změny v projektu, ale také pomocí jednoduchých příkazů umožňuje nahrát projekt z prostředí konzole vývojáře do repozitáře. Díky možnosti provádět pull requesty, což jsou žádosti o stažení verze aplikace, je GitHub vhodný pro větší projekty a rozsáhlejší týmy vývojářů. Práce s těmito funkcemi umožňuje paralelní vývoj aplikace, což je nezbytné u rozsáhlejších projektů. (11)

1.4.2 Railway

Railway je nástroj, který podporuje vývoj webových aplikací s důrazem na usnadnění fáze nasazení do produkce. Hlavním cílem je zjednodušit proces nasazení aplikace díky uživatelskému rozhraní, které poskytuje. Railway umožňuje nasazení pomocí funkce drag and drop, což znamená, že uživatelé mohou prostřednictvím jednoduchých kliků vytvořit vhodné prostředí, do kterého lze aplikaci nasadit. Díky této funkci nemusí vývojáři psát rozsáhlý kód, který by jinak byl potřebný pro nasazení. (12)

Railway rovněž podporuje manipulaci s GitHub repozitáři. GitHub umožňuje uživatelům nahrávat části aplikace do repozitářů, ze kterých může Railway převzít kód. Tato integrace usnadňuje nahrávání jednotlivých částí přímo do Railway aplikace a urychluje tak celý proces nasazení. Také sleduje jednotlivé repozitáře, ve kterých jsou nahrané části aplikace. Pokud dojde v repozitáři ke změně, Railway provede aktualizaci právě této části. Aplikace Railway je primárně zaměřena na zjednodušení procesu vývoje a nasazení webových aplikací, což umožňuje vývojářům rychleji a efektivněji uvést své aplikace do provozu. (11, 12)

1.4.3 Stripe

Stripe je platforma, která firmám umožňuje řídit online platební operace. Nabízí širokou škálu funkcí jako je zpracování plateb, správa zákaznických účtů, podpora různých platebních metod nebo zabezpečení platebních transakcí. Hlavním důvodem pro využití Stripe v aplikaci je právě jeho schopnost přijímat platby od uživatelů. Stripe nabízí možnost implementovat jejich API, což je rozhraní, které umožňuje aplikaci využívat vybrané funkce přímo v ní. Stripe také nabízí možnost testovacího módu, který mohou vývojáři využít pro nasazení rozhraní do aplikace. Poté, co je aplikace nasazena do produkce a uživatel má možnost provést platbu, Stripe si účtuje malý poplatek za každou přijatou transakci. (13)

Stripe také podporuje vytváření webhooků. Jedná se o funkce v aplikaci, které jsou volány po určité operaci. Tyto operace lze definovat přímo v aplikaci Stripe. Do projektu byl implementován webhook, který se aktivuje po provedení platby uživatele a následně umožní její uložení do systému. Tento proces je důležitý, protože bez potvrzení o platbě by mohla být nezaplacená nebo neúspěšná objednávka uložena do databáze a napáchat tak škody. (13)

1.5 Použité metody

Vývoj softwaru je proces, při kterém převádíme myšlenky zadavatele do hotového produktu. Tento proces ale nemusí být příliš snadný. Při vývoji rozsáhlejších aplikací se obvykle podílí tým, který člení a zastupuje jednotlivé funkce. Může se jednat o programátory jak přední strany aplikace, tak zadní, databázové specialisty, grafické designéry a mnoho dalších. V případě, že se na projektu podílí více lidí, může být obtížné sjednotit myšlenky a požadavky na aplikaci. Následující část popisuje postupy a metody, které jsou využívány pro zjednodušení vývoje softwaru.

1.5.1 UML

UML (Unified Modeling Language) je nástroj široce rozšířený v oblasti softwarového inženýrství. UML napomáhá k vizualizaci a následné konstrukci rozsáhlých systémů. Jeho hlavním cílem je zachycení a sjednocení myšlenek zadavatele pomocí diagramů a symbolů. Umožňuje předdefinovat jednotlivé části systému a jeho funkcionalitu, což napomáhá rychlosti vývoje a kvalitě systému. Existuje velké množství diagramů, které můžeme využívat, jako jsou například třídy, use case, sekvence, stavové a aktivitní diagramy, které nám umožňují navrhovat systémy a definovat jednotlivé interakce mezi nimi. (14)

V projektu bylo UML převážně použito k definování funkcionalit jednotlivých rozhraní aplikace. Díky dostupným nástrojům jako je například draw.io, je manipulace s diagramy velmi snadná a příjemná. Ne vždy musí zadavatel mít znalosti v oblasti tvorby aplikací a díky jeho neznalosti může být obtížné přesně určit požadavky na aplikaci. S zadavatelem probíhaly konzultace ohledně jeho požadavků, které byly následně převedeny do use case diagramů. Před kompletním definováním funkcionalit mohl být diagram jednoduše pozměněn, pokud se zadavatel rozhodl rozšířit nebo snížit funkcionalitu aplikace.

1.5.2 ERD

ERD (Entity-Relationship Diagram) je grafický nástroj, který pomáhá vytvářet databázové systémy. Jeho hlavním úkolem je vizualizovat jednotlivé entity (tabulky) a vztahy mezi nimi. Entity mohou představovat například uživatele, sklady nebo produkty. ERD diagramy pomáhají vývojářům vizualizovat a porozumět struktuře databázového systému a zjednodušují jeho následnou implementaci. Díky tomu, že diagram obsahuje ucelenou myšlenku a celý rozsah potřebných entit, snižuje se pravděpodobnost, že bude databáze v průběhu vývoje upravována. To by mohlo vést k časovým prodlevám a finanční ztrátě společnosti. Diagramy tohoto typu usnadňují návrh rozsáhlých databází a optimalizují jejich efektivnost. (15)

1.5.3 MVC

MVC (Model-View-Controller) je postup modelování softwarových aplikací, který napomáhá vývoji modulárních a škálovatelných aplikací. MVC se skládá ze tří částí: model, pohled (view) a řadič (controller). Každá část má za úkol zpracovávat jednotlivé části aplikace. Hlavním úkolem modelu je manipulace s daty a obsluha logiky aplikace. Řadič funguje jako prostředník, přijímá požadavky od uživatele a následně podle nich rozhoduje o dalších krocích. Při tomto procesu komunikuje jak s modelem, tak i s pohledem. Pohled se stará o vizualizaci dat a rozhoduje, jaká data jsou zobrazována uživatelům. Použití MVC architektury může vést ke zlepšení organizace struktury kódu, umožňuje paralelní vývoj jednotlivých částí a celkově napomáhá údržbě a rozvoji aplikace. (16)

Aplikace převzala architekturu MVC. Není pevně definováná jak v kódu, tak v struktuře složek, ve kterých je kód uložen, a spíše následuje tento definovaný postup pro lepší udržitelnost kódu. Model je zastoupen převážně SQL dotazy prováděnými nad databází. Jedná se například o získávání dat či kontrolu údajů uživatele. Řadič po obdržení uživatelových požadavků rozhoduje o dalších akcích. Následně může požádat modul o provedení akce nebo v případě neodpovídajících údajů zaslat chybovou zprávu přímo do

pohledu. Pohled je v aplikaci zastoupen knihovnou React, která díky převzatým informacím rozhoduje, jaká data budou uživateli zobrazena. (16)

2 Analýza současného stavu

V následující kapitole se věnuji analýze současného stavu společnosti Joynda s.r.o. Na začátku popisuji samotnou strukturu společnosti, včetně jejích podnikatelských plánů a potenciální konkurence. Dále se zabývám samotnou aplikací, jejím návrhem a specifickými požadavky na ni.

2.1 Představení společnosti

Joynda s.r.o. je začínající společnost, která se věnuje výrobě převážně oblečení a módních doplňků. V současné době firma aktivně nepůsobí na trhu, protože čeká až bude aplikace v dobrém stavu, aby mohla začít nabízet své produkty. V tuto chvíli firma zaměstnává 3 zaměstnance, mezi nimiž je i grafický designer, kterému bude následně aplikace předána, aby mohl vytvořit její ucelený vzhled. Firma má v plánu rozšiřovat své působení jak na trhu, tak i co se týče sortimentu produktů. V prvních fázích firma hodlá nabízet své designy, převážně formou potisku na trička, které nechává následně vytvářet u třetí strany. V dalších fázích chce firma expandovat a rozšířit svůj sortiment o doplňky, obuv, nábytek a také plánuje skladovat vlastní zásoby, aby snížila cenu produktů.

2.2 Požadavky na aplikaci

Požadavky na aplikaci byly dopředu stanoveny majitelem firmy. Aplikace v průběhu vývoje byla konzultována s majitelem, kvůli specifikaci jeho vize, a upravována podle potřeb, než vznikl produkt, který splňoval jeho představu. Hlavními požadavky byly různá uživatelská rozhraní a jejich funkce. Aplikace musí obsahovat rozhraní pro administrátora, které umožní manipulaci se záznamy, a rozhraní uživatele, které umozní nákup produktů a díky kterému si bude moci procházet své uložené záznamy.

Administrátorské rozhraní

Aplikace musí obsahovat administrátorské rozhraní, díky kterému bude administrátor schopen přidávat kolekce a produkty a následně na nich provádět úpravy, případně mít možnost smazání záznamu. Administrátor dále musí být schopen manipulovat s objednávkami a reklamacemi od zákazníků.

Rozhraní přihlášeného uživatele

Rozhraní pro přihlášeného uživatele musí umožňovat vytvářet reklamace, možnost změny hesla, ukazovat předchozí objednávky, přidávat, upravovat a mazat uživatelské údaje k doručení produktů a zobrazovat zadání reklamace.

Rozhraní nepřihlašeného uživatele

Nepřihlášený uživatel musí mít možnost provést reklamaci nebo změnu hesla, pokud se daný záznam vyskytuje v systému, a stejně jako u předchozích rozhraní, uživatel musí mít možnost zakoupit vybrané produkty.

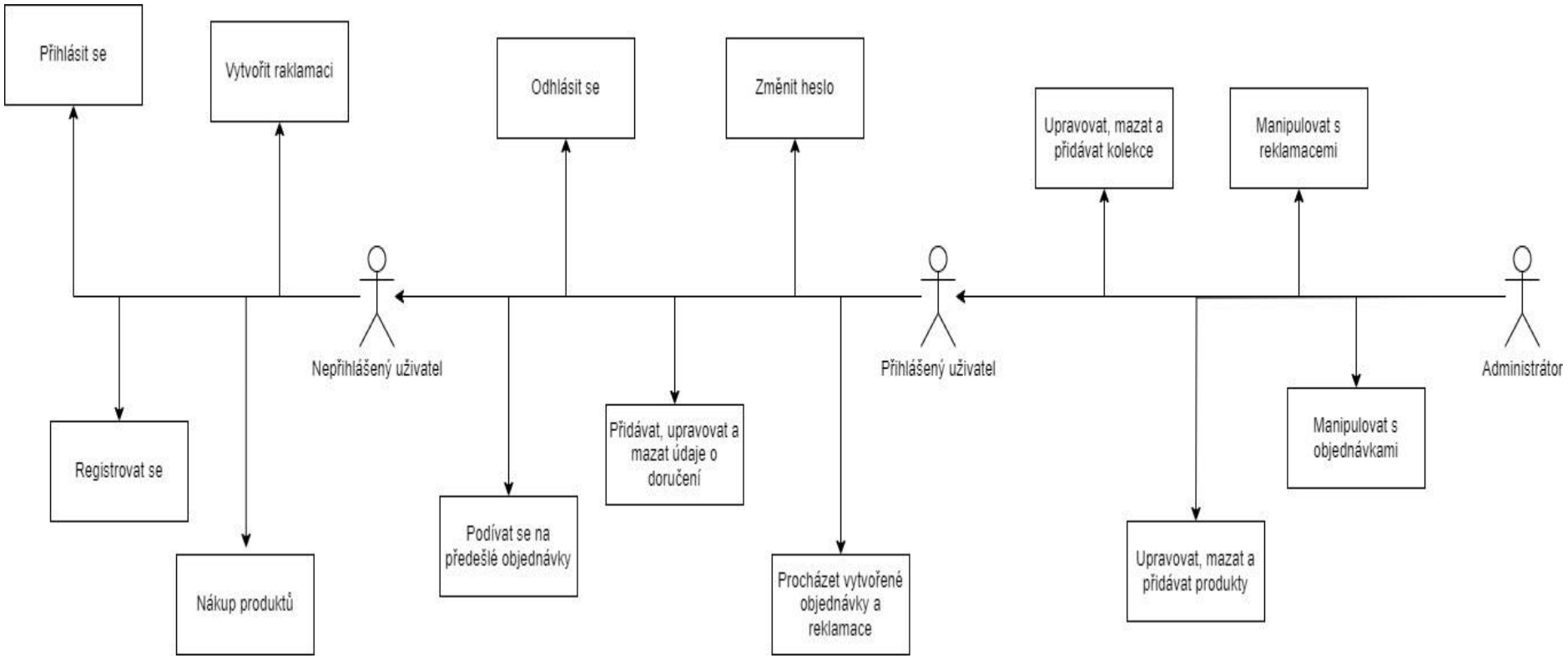
2.3 Návrh aplikace

Aplikace byla vyvinuta primárně pro aktuální potřeby firmy, tedy první fázi, která byla zmíněna výše. Při návrhu aplikace jsem se snažil brát v potaz možné prvky, které by firma mohla požadovat do budoucna, a implementovat je do aplikace. Díky tomu, že se jedná o začínající firmu, a majitel v tuto chvíli nemá ucelenou představu o budoucnosti firmy, byl tento krok poměrně obtížný. Snažil jsem se tedy převážně vytvořit dobrou strukturu kódu, aby aplikace mohla být jednoduše rozšířena a jednotlivé aspekty upraveny podle potřeb. Před samotnou tvorbou aplikace je potřeba si ujasnit, co přesně od aplikace požadujeme a jak toho dosáhneme. Navrhování struktury a funkcionality je důležité z hlediska času a financí. Zadavatel nemusí mít přehled o postupech používaných pro

tvorbu webových aplikací, a proto je důležité, abychom plně porozumněli jeho vizi pro následnou tvorbu.

Use Case diagram

Use Case diagram (Diagram případů užití) je nástroj používaný pro zjednodušení návrhu a implementace projektu. Pomáhá vizuálně interpretovat požadavky zadavatele a na jednom místě sjednotit celou funkcionalitu aplikace. V následujícím diagramu jsou zobrazena jednotlivá rozhraní požadovaná zadavatelem a jejich funkce. (14)



Obr. 3 Use Case diagram

Zdroj: Vytvořeno autorem bakalářské práce

2.4 Porovnaní

Na trhu existuje velké množství firem, které se zabývají prodejem oblečení a doplňků. Dají se rozdělit do dvou skupin. Jedna jsou firmy, které míří na masovou výrobu a jejich cílem je prodat co nejvíce kusů a nehledí příliš na to, jak se prezentují nebo jaké produktu kumu prodávají. Druhá skupina se zaměřuje spíše na kvalitu produktu a snaží se vést svou značku jako jakýsi životní styl. V této části srovnávám tuto aplikaci s konkurencí z obou sektorů.

Masová výroba

Asos.com je firma pocházející z Anglie, která se zabývá masovým prodejem oblečení a doplňků. Firma nabízí široký sortiment produktů a očekává vysokou návštěvnost, což znamená, že aplikace musí být dobře optimalizována, aby nenastávaly problémy s načítáním produktů, a musí obsahovat rozsáhlé možnosti třídění, aby uživatel mohl jednoduše nalézt, co právě hledá. Aplikace má také zabudovanou možnost provést reklamaci přímo na stránce a nemusí tedy ručně provádět a prověřovat, zda zákazník má nárok na reklamaci, čímž šetří čas i peníze. Asos perfektně zvládá všechny zmíněné body, má dobře optimalizované načítání stránky, rozsáhlé možnosti filtrace u všech druhů produktů a kvalitní reklamační systém.

Lifestyle

Supreme.com je původně skateboardová značka původem z USA, v dnešní době se spíše zabývá prodejem oblečení a upomínkových předmětů, jako jsou popelníky, cihly nebo páčidla. Firma se prezentuje jako více luxusní a unikátní než například Asos. Produkty, které nabízejí, jsou většinou limitované a tím se zvyšuje poptávka po nich. Firma dbá hlavně na kvalitu a je si vědoma, že její jméno ve světě něco znamená. Existuje spousta fanoušků, kteří touto značkou žijí. Toto se odráží i na jejich stránkách. Stránky jsou poměrně minimalistické a nenabízejí příliš informací o produktech.

Porovnání s aplikaci

Aplikace si něco vzala z obou světů. Společnost se spíše přiklání k "lifestyle" vizi, jelikož firma nemá v tuto chvíli v plánu provádět masový prodej, ale spíše si vybudovat jméno a klientelu. V budoucnosti je potřeba počítat s růstem, proto i tato myšlenka byla zapojena při navrhování aplikace. V aplikaci nenaleznete příliš mnoho možností na třídění, diky tomu že firma nemá v plánu produkovat velké množství různých designů a produktů. Uživatel může produkty třídit pomocí kolekcí nebo jména produktu. Velký důraz byl kladen na uživatelské rozhraní, které umožňuje uživateli provádět spoustu úkonů sám, například provádět reklamace, díky čemuž je aplikace poměrně v dobrém stavu co se týče budoucnosti a automatizace.

3 Vlastní návrhy řešení

V následující části popisuji technické řešení projektu, jeho jednotlivé části, metody a postupy využité při vývoji.

3.1 Databáze

Databáze je nedílnou součástí téměř každé aplikace. Má za úkol ukládat data, jako jsou například přihlašovací údaje uživatele nebo informace o objednávkách či produktech. Databáze ukládá data, která mohou být dále použita pro analýzu, ze které firma může získávat cenné informace, jakožto informace o cílových skupinách zákazníků, které na jejich stránkách nakupují nejvíce, který produkt je nejprodávanější a tak dále. Díky těmto informacím mohou firmy upravit své strategie a optimalizovat zisk. Ač jsou tyto informace velmi cenným prvkem, v tomto projektu je hlavním využitím databáze ukládání dat a následná možnost jejich získání, jež může být zobrazeno na stránce. (2)

Webové aplikace samy o sobě mají možnost v sobě uchovávat data, ať už se jedná o struktury zasazené přímo v kódu, nebo prostředky jako jsou session storage či cookies. Tyto prostředky jsou velmi užitečné a v dnešní době často používané, ale mají i své nevýhody. Pokud bychom chtěli ukládat data přímo do kódu, neměli bychom možnost je jakkoliv upravovat ze stránky a mohli bychom tím pouze vytvořit statickou stránku. Co se týče zmíněného session storage nebo local storage, tyto úložiště jsou přímo na stránce a poskytují i možnost manipulace s nimi. To znamená že díky nim můžeme manipulovat i s obsahem stránky. Tyto úložiště nejsou ideální pro větší aplikace, mají velmi omezenou kapacitu a uživatel s nimi může volně manipulovat pomocí konzole v prohlížeči. Výhodou aplikací které nepoužívají databázi, je vyšší rychlost. Aplikace se nemusí připojovat k databázi aby získala data, což u rozsáhlejších dotazů může mít za následek pomalejší odpověď ze strany serveru. (17)

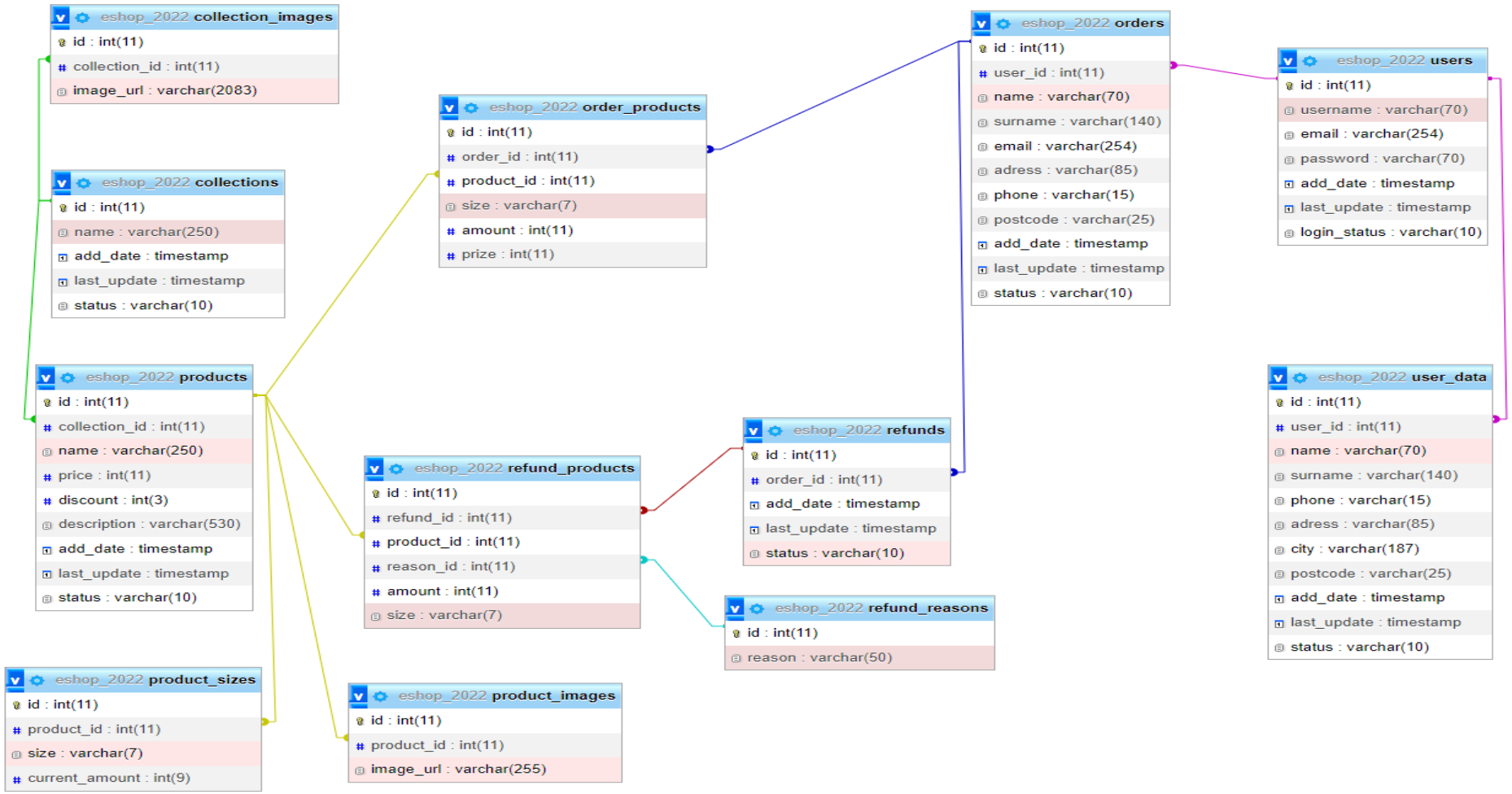
V dnešní době existuje několik možností, jak ukládat data, ale mezi hlavní patří SQL a NoSQL databáze. Rozdíl mezi nimi spočívá v procesu, jak data ukládají, a jak udržují vztahy mezi záznamy. SQL databáze ukládají data do tabulek, složených z řádků a

sloupců, kde se data mohou odkazovat a propojovat, a tím mezi sebou vytvářet vztahy. Na druhou stranu NoSQL databáze ukládají data do JSON souborů ve formě schémat, což je ekvivalent SQL tabulek, s tím rozdílem, že zde nejsou zachyceny vazby mezi jednotlivými záznamy. Tento postup je dobrý pro blogy nebo aplikace podobného typu, které chtějí skladovat velké množství záznamů. Jelikož jsou data v aplikaci úzce propojena, ať už se jedná o záznamy uživatele nebo produkty samotné, rozhodl jsem se v projektu použít SQL databázi, konkrétně MySQL databázi. Pro zjednodušení práce nad databází jsem zvolil rozhraní phpMyAdmin, které poskytuje snadné a přehledné pracovní prostředí. (2, 3, 5)

3.1.1 Návrh databáze

Správně navržená databáze je klíčem k efektivnímu a úspěšnému fungování aplikace. K tomu slouží návrh databáze pomocí ERD diagramu (Entity-Relationship Diagram). Správně navržená databáze by měla nejen být optimalizovaná pro rychlý přístup k datům, ale měla by také zohledňovat budoucí změny v podnikatelském plánu a potřeby zadavatele. (15)

Při konzultacích se zadavatelem jsem se dozvěděl, že jeho podnikatelský plán se bude měnit, a tím pádem pravděpodobně i databáze sama o sobě. Snažili jste se tuto skutečnost brát v potaz při návrhu databáze, aby nebyly nutné velké úpravy v budoucnosti. Nicméně budoucí potřeby majitele se mohou měnit, a proto je důležité navrhnout databázi flexibilně a s možností rozšíření či změn v budoucnu.

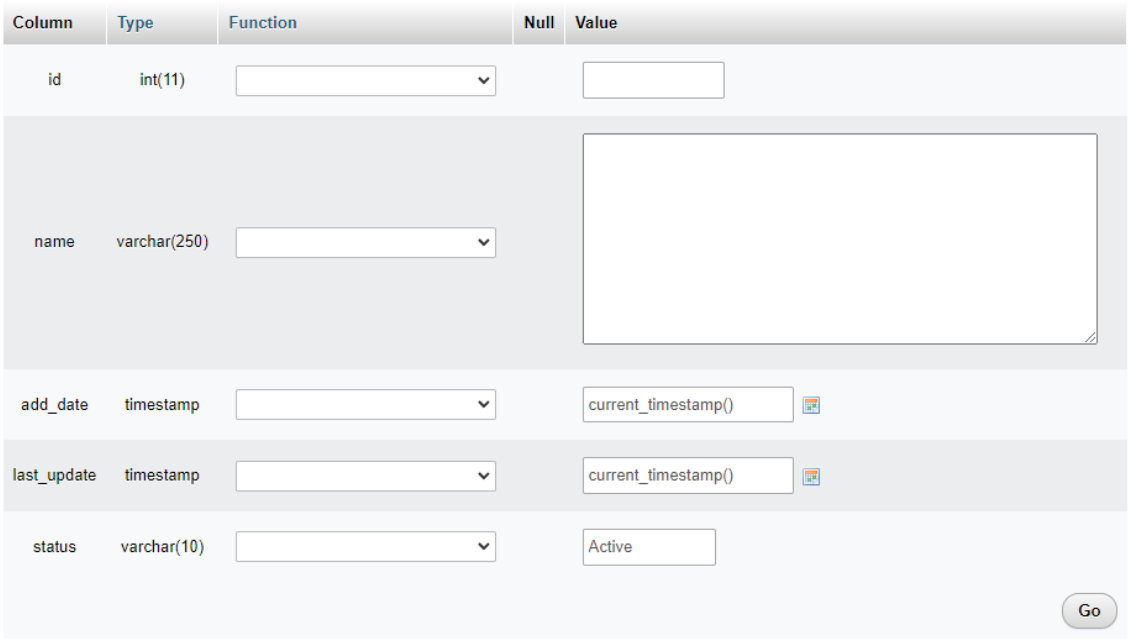


Obr. 4 ERD diagram

Zdroj: Aplikace phpMyAdmin

3.1.2 Ukázka phpMyAdmin

PhpMyAdmin je rozhraní, které nám umožňuje jednodušší přístup a manipulaci s databází. Díky němu nemusíme obsluhovat databázi pomocí konzole, ale můžeme s ní manipulovat pomocí aplikace. Toto vytváří příjemné pracovní prostředí, ve kterém přehledně vidíme stav databáze, tabulek a mnoho dalšího. Podpora takzvaného "klikacího" rozhraní usnadňuje a urychluje tvorbu databáze.



Obr. 5 Ukázka rozhraní phpMyAdmin pro přidávání záznamů

Zdroj: Aplikace phpMyAdmin

3.2 Backend

Backend referuje „zadní“ stranu aplikace. Jedná se o část aplikace, která má za úkol obsluhovat logiku a splňovat požadavky od uživatele. Za celou logiku aplikace se spoléhá Node.js společně s Express frameworkem. Express je zodpovědný za vedení serveru a obsluhování požadavků na něj. (17)

3.2.1 Požadavky na server

Úkolem serveru je zpracovávat požadavky od uživatele a následně vracet odpovědi. Může se jednat například o data z databáze nebo v případě neúspěchu o chybovou hlášku. Server zpracovává požadavky pomocí tzv. tras (routes), což jsou cesty ve kterých je přesně definováno, jak má postupovat. Tyto trasy jsou definovány jak na straně uživatele, tak na straně serveru. Pokud uživatel zašle požadavek na server, bude vyhodnocen a přidělen adekvátní trase, kde následně provede nezbytné úkony. (10, 17)

3.2.2 Start serveru

Před tím, než uživatel může aplikaci využívat, je potřeba vytvořit prostředí, kde budou vyslyšeny jeho požadavky. V našem případě to znamená aktivaci serveru, který tyto požadavky přijme. Server je možné si upravit podle potřeb, a také přidávat knihovny, které rozšíří schopnosti serveru. Například knihovna fileupload nám umožňuje manipulaci se soubory. V aplikaci je použita na ukládání fotek produktů na server.

Server se musí také spustit na určitém portu, což funguje jako ukazatel, kde se do aplikace lze připojit. Je to důležité jak pro produkční, tak testovací fázi projektu. V testovací fázi se odkazujeme na localhost a port, aby server mohl vyslechnout naše požadavky. V produkční fázi je důležité do samotné aplikace Railway uvést port, na kterém bude server spuštěn, aby aplikace dokázala správně fungovat. (17)

```
import express, {Express} from "express";
import cors from 'cors';
import fileUpload from 'express-fileupload'
import bp from 'body-parser';
import {router} from './routes.js';

require("dotenv").config()

const app: Express = express();

app.use(cors({
  // origin: 'https://nodejs-production-ff30.up.railway.app' || 'http://localhost:8001'
}))   //enable the express server to respond to preflight requests

app.use(bp.urlencoded({extended: true}));   //support parsing of application/x-www-form-urlencoded post data
app.use(bp.json())    // support parsing of application/json type post data
app.use(fileUpload())   //file support

app.use(express.static('public')) //udeluje pristup k server public dir

app.use('/', router)  // pristupuje k app.post/get requestum

const port = process.env.PORT || 4000;

app.listen(port, function () {
  console.log("server on port " + port)
});
```

Obr. 6 Zdrojový kód pro nastartování serveru

Zdroj: Vytvořeno autorem bakalářské práce

3.2.3 Middleware

Middleware je označení pro funkce, které proběhnou před tím než server odešle odpověď uživateli. Jejich úkolem je zjednodušit a zpřehlednit aplikaci. Zmiňované trasy „routes“ jsou také považovány za middleware díky podobné manipulaci dat, přičemž hlavním rozdílem je jejich volání. Router-level middleware je volán pro konkrétní cesty, jako například /add_record v projektu, zatímco Application-level middleware je obvykle volán v Router-level middleware před jeho provedením. Pro následující část jsem zvolil několik příkladů middleware a routes a podrobně je popisuji. (10, 17)

Tabulka 1 Seznam middleware použitého v aplikaci

Název	Popis
request_data_transformer	Transformuje data do příjemnějšího tvaru
login_request_validation	Kontroluje, zda uživatel splňuje požadavky k přihlášení
validate_user_data	Kontroluje, zda uživatel existuje
register_request_validation	Kontroluje, zda uživatel splňuje požadavky pro registraci
refund_request_validation	Kontroluje, zda uživatel splňuje požadavky pro položení reklamace
check_for_duplicit_record	Kontroluje, zda není záznam v databázi

Zdroj: Vytvořeno autorem bakalářské práce

Check for duplicit record

Provádí jednoduchý dotaz nad databází, aby zjistil, jestli se v ní záznam nachází, a podle typu záznamu provede následující akce: buď zavolá funkci "next", a aplikace postoupí na další middleware, nebo pošle chybovou zprávu a proces se ukončí.

```
const result = await select_request(sql, data.wheres.values)

if(req.body.refund){
    if(result.length <= 0){
        return next(new Error("incorect email or order id"))
    }else{
        req.body.order_data = result
        return next()
    }
}else if(req.body.auth_code){
    if(result.length <= 0){
        return next(new Error("user not found"))
    }else{
        req.body.user_id_auth = result[0].id
        return next()
    }
}else if(req.body.psw_change || req.body.order){
    return next()
}else{
    if(result.length > 0){
        return next(new Error("record already in system"))
    }
}

next()
```

Obr. 7 Zdrojový kód pro kontrolu duplicity

Zdroj: Vytvořeno autorem bakalářské práce

Validate user data

Díky tomu, že data uživatele jsou uložena v session storage je možné, že by k nim uživatel mohl získat přístup a změnit je. Pokud by k takové situaci došlo, existuje šance, že by se mohl dostat k informacím ostatních uživatelů, což není žádoucí. Tato funkce tomu zabrání tím, že vezme data ze session storage a ověří, jestli uživatel existuje a jestli se shoduje jeho zadaný email a heslo. Někdo by mohl namítat, že pokud by uživatel zadal email a heslo jiného uživatele, mohl by se stále dostat k jeho informacím. Toho stejného by však dosáhl i pokud by tyto informace zadával do přihlašovacího formuláře, takže by musel znát email a heslo jiného uživatele.

```
import { Request, Response, NextFunction } from "express";
import select_request from "../../DB/select_request";

export default async function validate_user_data(req: Request, res: Response, next: NextFunction){

    const email = JSON.parse(req.body.email)
    const password = JSON.parse(req.body.password)

    const sql = "SELECT * FROM users WHERE email = ? && password = ?;"

    const result = await select_request(sql, [email, password])

    if(result.length <= 0){
       return next(new Error("user is not in system"))
    }else{
        req.body.user_data = result[0]
    }

    next()
}
```

Obr. 8 Zdrojový kód pro kontrolu dat uživatele

Zdroj: Vytvořeno autorem bakalářské práce

Login request validation

Kontroluje, zda je uživatel v systému a zda jsou data, která zadal do přihlašovacího formuláře správná, tedy zda účet přihlášený pod určitým emailem má dané heslo. Díky tomu, že hesla jsou v databázi zašifrována, je potřeba je dešifrovat, abychom je mohli porovnat. Nicméně zašifrovaná hesla nelze dešifrovat, protože by to narušilo celou ochranu. Místo toho používáme modul bcrypt, který tato hesla porovnává a vrátí nám odpověď.

```
const login_request_validation = async(req: Request, res: Response, next: NextFunction) => {

    var email_test_result: any = await select_request("SELECT * FROM users WHERE email = ? ;",

    if(email_test_result.length <= 0){
        return next(new Error("email not in system"))
    }

    var password_in_db = email_test_result[0].password

    const data = req.body.transformed_data

    var is_match = await bcrypt.compare(data.password, password_in_db)

    if(!is_match){
        return next(new Error("wrong password"))
    }

    req.body.login_request_validation = {msg: "correct data for login", next_status: true,

    next()
}
```

Obr. 9 Zdrojový kód pro kontrolu uživatele v systému

Zdroj: Vytvořeno autorem bakalářské práce

Tabulka 2 Seznam routes použitého v aplikaci

Název	Popis
/webhook	Umožňuje platbu pomocí Stripe API
/stripe_create_session	Generuje URL adresu pro platební portál
/validate_cart_items	Kontroluje, zda jsou všechna data v košíku nepoškozena
/login_request	Umožňuje uživateli se přihlásit
/logoff_request	Umožňuje uživateli se odhlásit
/register_request	Umožňuje uživateli se registrovat
/add_record	Přidá záznam
/edit_record	Upraví záznam
/change_record_status	Změní stav záznamu
/refund_request	Zkontroluje, zda je možné podat reklamaci
/send_aut_code	Pošle ověřovací kód na email
/get_user_data	Získá údaje uživatele
/main_page_request	Získá data pro hlavní stránku aplikace
/get_product_by_id	Najde produkt pomocí ID
/get_collections	Získá kolekce
/get_collections_showcase	Získá kolekce, které mají k nim přidané produkty
/get_collection_product_showcase	Získá data produktu podle zadaného ID
/get_refund_reasons	Získá důvody pro reklamace
/get_user_refunds	Získá reklamace uživatele
/get_user_acccount_data	Získá data o účtu uživatele
/get_user_avaible_returns	Získá všechny objednávky nad kterými lze provést reklamace
/get_user_place_returns	Získá všechny objednávky nad kterými byla provedena reklamace
/get_user_placed_orders	Získá objednávky uživatele
/get_admin_collections	Získá data o kolekci pro stránku administrátora
/get_admin_products	Získá data o produktech pro stránku administrátora
/get_admin_orders	Získá data o objednávkách
/get_admin_refunds	Získá data o reklamacích
/check_for_admin	Kontroluje, zda je uživatel administrátor

Zdroj: Vytvořeno autorem bakalářské práce

Add record

Pokud je na server poslán požadavek na trasu /add_record, aplikace se podívá, jestli se záznam již nevyskytuje v systému, a pokud ne, uloží ho do databáze. Jinak zahlásí chybovou hlášku. Požadavek s requestem může obsahovat i soubory (fotky), které server uloží a následně upraví.

```
const transformed_data = req.body.transformed_data

var record_id = await insert_records(transformed_data.tables, transformed_data.columns, transformed_data.values)

if(req.files){
  await save_files("./public/images/temp/", req.files)
  modify_images("./public/images/temp/", record_id, JSON.parse(req.body.folder))
}

res.send({msg: "record added", next_status: true, status: true})
```

Obr. 10 Zdrojový kód pro přidání záznamu do databáze

Zdroj: Vytvořeno autorem bakalářské práce

Get collections

Cesty s příponou get mají za úkol získávat data z databáze. Většinou se potřebují vytáhnout více dat k jednomu záznamu, ta jsou načtena a následně spojena do jednoho objektu, se kterým může pracovat přední strana aplikace.

```
router.post('/get_collections', try_catch(async function (req: Request, res: Response) {

  var data: any = await Promise.all([write_json(["SELECT collections.id, collections.name," +
  "DATE_FORMAT(collections.add_date, '%Y-%m-%d') as add_date, collection_images.image_url" +
  "FROM collections JOIN collection_images ON collection_images.collection_id = collections.id" +
  "WHERE collection_images.image_url LIKE '%_main%' AND collections.status = 'Active';",
  "SELECT collection_images.image_url FROM collection_images WHERE collection_images.collection_id = $"])])

  res.send(JSON.parse(data))

}))
```

Obr. 11 Zdrojový kód pro získání dat kolekcí

Zdroj: Vytvořeno autorem bakalářské práce

Edit record

Zajišťuje úpravu záznamu v databázi. Pokud projde kontrolou duplicity, následně upraví záznam v databázi. Vzhledem k tomu, o jaký záznam se jedná, je specificky upravuje. Například pokud má záznam k sobě přiřazené fotografie, je potřeba provést úpravu i těchto fotografií v databázi i na serveru.

```
const transformed_data = req.body.transformed_data

if(req.body.files_names_to_keep){
  await update_records(transformed_data.tables, transformed_data.columns, transformed_data.values,
    JSON.parse(req.body.record_id), JSON.parse(req.body.files_names_to_keep))
}else{
  await update_records(transformed_data.tables, transformed_data.columns, transformed_data.values,
    JSON.parse(req.body.record_id))
}

if(req.files){
  await update_files(JSON.parse(req.body.files_names_to_keep), JSON.parse(req.body.folder),
    JSON.parse(req.body.record_id), req.files)
  modify_images("./public/images/temp/", JSON.parse(req.body.record_id), JSON.parse(req.body.folder))
}else if(req.body.files_names_to_keep){
  await update_files(JSON.parse(req.body.files_names_to_keep), JSON.parse(req.body.folder),
  JSON.parse(req.body.record_id))
}

if(req.body.psw_change){
  return res.send({msg: "password changed", next_status: true})
}

res.send({msg: "record edited", next_status: true})
```

Obr. 12 Zdrojový kód pro editaci záznamů

Zdroj: Vytvořeno autorem bakalářské práce

Webhook

Je speciální middleware, který je volán třetí stranou po provedení určité akce. Stripe poskytuje podporu webhooků, je potřeba jej registrovat přímo na jejich stránkách a propojit s naší aplikací. Při vytváření webhooku je potřeba definovat, při kterých akcích chceme, aby byl spuštěn. V tomto případě je webhook aktivován, pokud uživatel provede platbu. Následná logika je definována v aplikaci. Pokud byla platba úspěšná, je objednávka uložena do systému a zákazníkovi je zaslán email s účtenkou. (18)

```
router.post('/webhook', express.raw({type: 'application/json'}), try_catch(async function (req: Request, res: Response) {

  const sig = req.headers['stripe-signature'];

  let data;
  let event_type;

  if(endpointSecret){

    let event;

    try {
      event = stripe.webhooks.constructEvent(req.body, sig, endpointSecret);
      console.log("webhook verified")
    } catch (err: any) {
      console.log("🚀 ~ router.post ~ err:", err.message)
      console.log("webhook not verified")
      res.status(400).send(`Webhook Error: ${err.message}`);
      return;
    }

    data = event.data.object
    event_type = event.type
  }else{
    data = req.body.data.object
    event_type = req.body.type
  }

  // Handle the event
  if(event_type === "checkout.session.completed"){
    var cunstomer_data = await stripe.customers.retrieve(data.customer)

    var transformed_data = JSON.parse(cunstomer_data.metadata.data)
    var cart_data = JSON.parse(cunstomer_data.metadata.cart)

    var order_id = await insert_records(transformed_data.tables, transformed_data.columns, transformed_data.values)

    send_receipt(transformed_data.email, JSON.parse(cart_data), order_id)
  }

  res.send().end;
}));
```

Obr. 13 Zdrojový kód pro Stripe webhook

Zdroj: Vytvořeno autorem bakalářské práce

Register request

Tento middleware má za úkol zpracovat požadavek na registraci od uživatele. Provede kontrolu, zda se v systému již nevyskytuje email, který uživatel zadal, a na základě toho provádí další akce. Pokud záznam nebyl nalezen, uživatel je registrován do systému, jinak je uživateli vypsána chybová hláška.

```
const transformed_data = req.body.transformed_data

const record_id = await insert_records(transformed_data.tables, transformed_data.columns,
  transformed_data.values)

const user_data = await select_request("SELECT id, username, email, password, login_status" +
  "FROM users WHERE id = ?", [record_id.toString()])

const user_account_data = await select_request("SELECT id, name, surname, phone, adress, city, postcode" +
  "FROM user_data WHERE user_id = ?", [record_id.toString()])

res.send({msg: "user registred", next_status: true, status: true, user_data: user_data,
user_account_data: user_account_data})
```

Obr. 14 Zdrojový kód pro registraci uživatele do systému

Zdroj: Vytvořeno autorem bakalářské práce

3.2.4 Ostatní funkce serveru

Hlavním úkolem serveru je splňovat požadavky, které kladou uživatelé. Jedná se o větší požadavky, které jsou plněny kombinací funkcí v aplikaci, a musí být řešeny přímo z určených tras. V následující části popisuji menší funkce, které server zastává.

Úprava a ukládaní fotografií

Jedním z úkolů serveru je ukládat fotografie na server. Aplikace pro tyto účely využívá modul fs (File System), který má v sobě zabudované funkce na manipulaci se soubory, jako jsou například mv pro přesunutí souboru do složky nebo readFile pro čtení souboru. Kapacita serveru není neomezená, a její navýšení zvyšuje náklady společnosti na údržbu aplikace. Proto je potřeba upravit ukládané soubory tak, aby jejich velikost co

nejméně ovlivňovala aplikaci. Úprava fotografií je prováděna modulem sharp, který napomáhá redukovat velikost fotografií uložených na serveru. Úprava velikosti fotografií je také důležitá pro přední část aplikace. Tato část získává fotografie přímo ze serveru, a jejich velikost může významně ovlivnit rychlost aplikace.

```
import * as fs from "fs"
import sharp from "sharp";

export default function modify_images(path: string, record_id: number, folder: string){

    var files = fs.readdirSync(path)

    if(!fs.existsSync("./public/images/" + folder + "/" + record_id)){
        fs.mkdirSync("./public/images/" + folder + "/" + record_id)
    }

    sharp.cache(false);

    for(let file of files){
        sharp(path + file)
        .jpeg()
        .jpeg({ quality: 50 })
        .jpeg({ progressive: true })
        //.resize(1920, 1080)  //HD pixels
        .toFile("./public/images/" + folder + "/" + record_id + "/" + file, (err: Error, info: any) => {
            if (err) {
            console.error(err);
            }else{
                fs.unlinkSync(path + file)
            }
        })
    }
}
```

Obr. 15 Zdrojový kód pro úpravu fotografií

Zdroj: Vytvořeno autorem bakalářské práce

Zasílaní emailů

Aplikace využívá modul Nodemailer pro zasílání důležitých emailů uživatelům. Tyto zprávy mohou obsahovat různé informace, jako jsou upozornění na slevy nebo potvrzení o přijetí objednávky. Hlavním využitím modulu Nodemailer je automatizované odesílání dokladů o zakoupení uživatelům po úspěšné platbě, což eliminuje potřebu fyzických faktur a zjednodušuje proces účtování pro firmu.

```
import * as nodemailer from "nodemailer"
import receipt_template from "./receipt_template";

export default function send_receipt(email: string, products: any, order_id: number){

    var transporter = nodemailer.createTransport({
        host: 'smtp.seznam.cz',
        port: 465,
        secure: true, // use SSL
        auth: {
            user: process.env.COMPANY_EMAIL,
            pass: process.env.COMPANY_EMAIL_PSW
        }
        });

        var html_template = receipt_template(products, order_id)

        var mailOptions = {
          from: process.env.COMPANY_EMAIL,
          to: email,
          subject: 'order receipt',
          html : html_template
        };

        transporter.sendMail(mailOptions, function(error, info){
          if (error) {
            console.log(error);
          } else {
            console.log('Email sent: ' + info.response);
          }
      })
}
```

Obr. 16 Zdrojový kód pro zasílání účtenek zákazníkovi

Zdroj: Vytvořeno autorem bakalářské práce

Dotazy na databázi

Jedním z nejdůležitějších úkolů serveru je komunikace s databází. Server nad databází provádí různé akce, jako jsou vybírání, editace nebo vkládání záznamu. Tato akce je rozdělena do dvou částí: příprava dotazu a samotné provedení.

Ukázka dotazu pro získávání záznamů

Pro získání potřebných dat potřebuje aplikace navázat spojení s databází. K tomu využívá tzv. pool systém, který jí přiděluje připojení, skrze které může komunikovat s databází. Po úspěšném navázání spojení je proveden požadovaný dotaz, získané informace jsou vráceny a následně je ukončeno spojení s databází, čímž se uvolní zdroje pro další připojení.

```
export default function select_request(sql: string, values?: Array<string>){

    return new Promise<Array<{id: number, [index: number]: Array<string>}>>((resolve, reject) => {

        pool.getConnection((conn_err: any, conn: any) => {
            if(conn_err){
                console.log("select_request; 🚀 ~ pool.getConnection ~ conn_err:", conn_err.message)
            }else{
                conn.query(sql, values, (err: any, result: any) => {
                    conn.release();
                    if(err){
                        console.log("select_request; 🚀 ~ conn.query ~ err:", err.message)
                    }else{
                        resolve(result)
                    }
                })
            }
        })
    })
```

Obr. 17 Zdrojový kód pro SQL dotaz

Zdroj: Vytvořeno autorem bakalářské práce

3.3 Bezpečnost

Webové aplikace jsou v dnešní době velmi užitečným nástrojem, který umožňuje jednoduchý přístup k informacím a produktům z celého světa. Nicméně je důležité mít na paměti, že tyto aplikace mohou být cílem útočníků, kteří se snaží poškodit nebo získat přístup k citlivým informacím. Aplikace tohoto typu mohou obsahovat cenné a citlivé údaje, které mohou lákat útočníky za účelem získání finančního prospěchu nebo poškození pověsti firmy. Proto je důležité tyto cílové objekty řádně zabezpečit.

SQL Injection

SQL Injection (SQLi) je technika útoku na webové aplikace, která využívá nedostatečného ošetření uživatelských vstupů ve formulářích nebo URL parametrech. Útočník vloží SQL kód do těchto vstupů, což může vést k neautorizovanému přístupu k databázi a manipulaci s jejím obsahem. (19)

Ošetření SQL Injection v aplikaci

Ošetření formulářů uživatele je klíčovým krokem pro zajištění bezpečnosti aplikace. Každý formulář má určitý datový typ, který může přijímat, jako jsou čísla, soubory nebo text. Tento krok poskytuje částečné zabezpečení tím, že definuje, jaká data může uživatel zadávat. Nicméně tento samotný krok nestačí, uživatelé mohou v textových polích formulářů vložit SQL dotazy, které mohou být spuštěny v databázi a tím poškodit její integritu. Aby se tomu předešlo, je důležité použít techniky jako jsou například parametrizované dotazy (Prepared Statements) a kontrola vstupních dat na straně serveru. Tyto opatření minimalizují riziko útoků typu SQL injection. (19)

SQL dotazy jsou posílány do aplikace jako řetězce znaků. Pokud by uživatel přidal vlastní dotaz na konec řetězce, byl by okamžitě vykonán a mohl by tak poškodit databázi. Tomuto lze jednoduše zabránit. Aplikace využívá funkci createPool z modulu mysql2, která umožňuje nastavení parametru multipleStatements. Pokud tento parametr nastavíme na

false, zajistíme, že se vykoná pouze první dotaz v řetězci, tedy ten náš. To má za následek, že uživatel nemůže vložit další dotazy, které by mohly potenciálně poškodit databázi.

Při provádění dotazů aplikace používá metodu nazývanou "Prepared statements". Tato metoda spočívá v tom, že databáze předem připraví SQL dotaz, do kterého jsou následně vkládána data uživatele. Jednou z výhod této metody je oddělení SQL dotazu a uživatelských dat, čímž se výrazně snižuje riziko SQL injection. To znamená, že i když uživatel vloží data obsahující SQL kód, databáze je vnímá jako data a nikoliv jako část SQL dotazu, čímž se minimalizuje riziko narušení integrity databáze. (19)

Ochrana uživatelského hesla

Hesla jsou obecně velmi citlivé informace. Pokud někdo získá naše heslo, má ve většině případů přístup k celému účtu uživatele a může ho zneužít. Uživatelská hesla jsou ukládána v databázi, což znamená, že pokud by se neoprávněná osoba dostala k těmto údajům, může je snadno získat a zneužít. Toto lze řešit pomocí šifrování hesel.

Pro šifrování hesel v naší aplikaci byl použit modul bcrypt, který poskytuje funkci hash. Tato funkce převede hesla do nečitelné podoby a uloží je takto zašifrovaná do databáze. I když jsou hesla stále uložena v databázi, jsou nečitelná a tím pádem se snižuje riziko úniku.

Bcrypt je efektivní nástroj v tom, že každé šifrování je unikátní. To znamená, že i když dva uživatelé zvolí stejná hesla, uložená hash hodnota bude pro každého z nich odlišná. To je důležité, protože to brání tomu, aby útočník na základě znalosti jednoho hashu odhadl hash hesla jiného uživatele.

Ochrana produktů v košíku

Produkty v košíku jsou v aplikaci ukládány pomocí session storage, ke kterému má uživatel přístup pomocí konzole v prohlížeči. Pokud by uživatel změnil parametry produktu, mohlo by to vést k chybám v systému a potenciálním krádežím. Toto je ošetřeno

jednoduchým dotazem, který kontroluje, zda se produkty v košíku shodují s těmi uloženými v databázi. Pokud ne, uživatel je informován, že prováděl neoprávněné akce, pomocí chybové hlášky. (17)

Ověřování reklamací a změny hesel

Uživatel má možnost změnit si heslo a podat reklamaci i v případě, že není přihlášen. Abychom zabránili manipulaci s účtem jiného uživatele, vyžadujeme zadání e-mailové adresy, na kterou je zaslán ověřovací kód. Po zadání správného kódu můžeme předpokládat, že se jedná o oprávněného uživatele a umožníme mu přístup k dané funkci.

3.4 Frontend

Frontend označuje „přední“ stranu aplikace, což je část aplikace, která je zodpovědná za vyobrazení dat. Přední část je reprezentována frameworkem React, který používá komponenty a stavy pro tvorbu dynamického webu. React využívá hooky jako jsou useEffect a useState k sledování stavu a změn stavu jednotlivých komponentů aplikace. Pokud je stav změněn, React aktualizuje pouze daný komponent bez nutnosti aktualizace celé stránky. Toto má za následek efektivnější aplikaci a usnadňuje tvorbu dynamických webových aplikací. (9)

3.4.1 Komponenty

Komponenty jsou jedním z nástrojů z arzenálu reaktu, které napomáhají při vývoji aplikací, zvyšují přehlednost kódu a rychlost vývoje aplikace. Komponenty jsou znovupoužitelné části kódu, které si můžeme představit jako funkce, jež například místo čísla vrací HTML elementy, které mohou být umístěny kamkoliv na stránce. Komponenty, stejně jako funkce, přijímají parametry, a jedná se o takzvané props, které mohou poskytovat komponentě data, funkce nebo jiné potřebné parametry pro správné fungování. Pro následující část jsem zvolil několik rozsáhlejších komponentů aplikace a popisuji je. (9)

Tabulka 3 Seznam komponentů v aplikaci

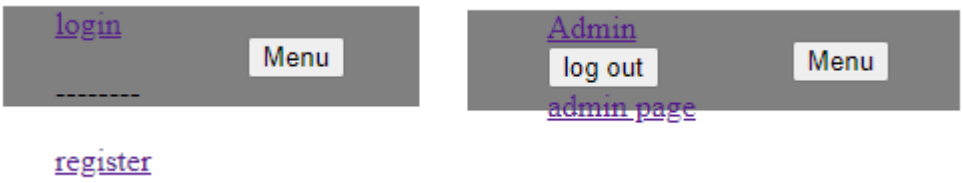
Název	Popis
User_size_select	Umožňuje uživateli vybírat velikosti produktu
User_show_case	Vyobrazuje fotografie produktu
Refund_table_row	Tabulka pro provedení reklamace
Product	Šablona produktu
Money_sum	Shrnutí ceny objednávky
Menu	Vysouvací menu s odkazy na další stránky
Login_hud	Ukazuje informace o stavu přihlášení

Loading	Ukazuje, kdy se načítají informace na stránce
Header	Záhlaví
Collection	Šablona kolekce
Cart	Ikona košíku informující uživatele o množství produktu, které plánuje zakoupit
Cart_items	Ukazuje produkty v košíku
Admin_add_image_handler	Rozhraní administrátora pro přidávání fotografií
Sub_image_add	Podpůrný komponent Admin_add_image_handler, který rozšiřuje funkcionalitu
Admin_collection_select	Rozhraní administrátora pro přidávání kolekcí
Admin_size_checkboxes	Rozhraní administrátora pro přidávání velikostí produktu

Zdroj: Vytvořeno autorem bakalářské práce

Login hud

Login hub se nachází v pravé horní části aplikace a má za úkol signalizovat uživateli, zda je přihlášen v systému nebo nikoliv. Pokud je uživatel přihlášen jako administrátor, login hub mu zpřístupní odkaz na stránku administrátora. Pokud je uživatel přihlášen jen jako zákazník, může se přes své uživatelské jméno dostat do nastavení účtu.

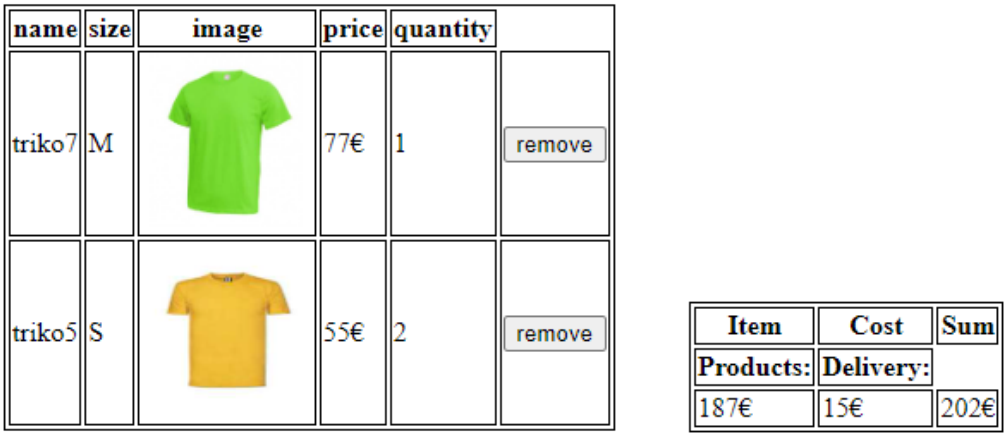


Obr. 18 Ukázka komponentu Login hud

Zdroj: Vytvořeno autorem bakalářské práce

Cart item & Money sub

Tyto komponenty mají za úkol signalizovat uživateli, jaké předměty si hodlá zakoupit, jejich velikost, cenu a množství. Pokud uživatel nadále nemá o produkty zájem, může je odebrat z košíku pomocí tlačítka "remove".

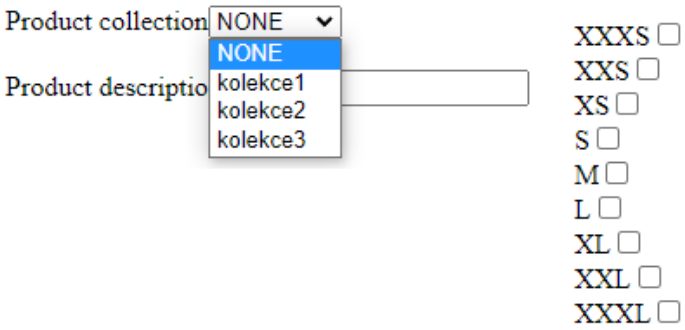


Obr. 19 Ukázka komponentu Cart item a Money sub

Zdroj: Vytvořeno autorem bakalářské práce

Admin collection select & Admin size checkboxes

Tyto komponenty mají za úkol vybrat vhodnou kolekci která je v systému, a následně vybrat velikost a množství produktu, jež jsou přidávány do systému.

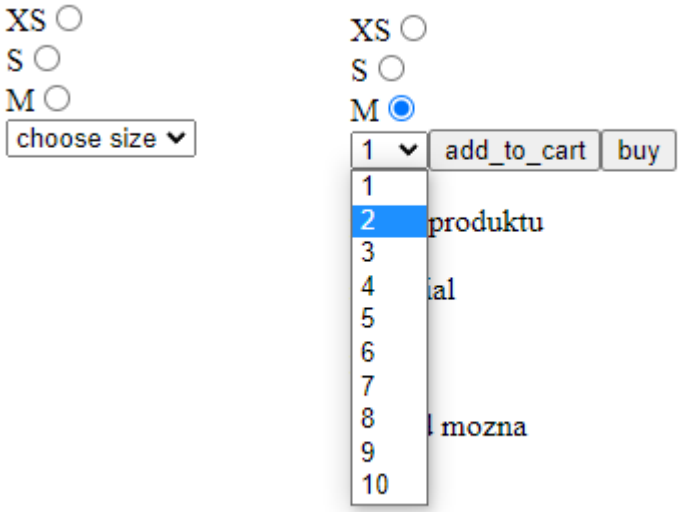


Obr. 20 Ukázka komponentů Admin collection select a Admin size checkboxes

Zdroj: Vytvořeno autorem bakalářské práce

User size select

Tento komponent zajišťuje, že do košíku není přidán produkt, u kterého není určena velikost a množství. Pokud byla velikost vybrána, uživatel může buď produkt přidat do košíku, nebo rovnou přejít na stránku, kde může pokračovat k platbě

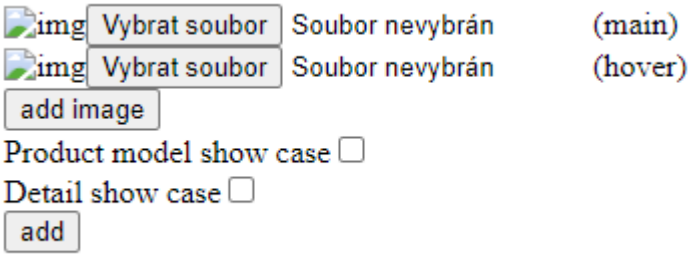


Obr. 21 Ukázka komponentu User size select

Zdroj: Vytvořeno autorem bakalářské práce

Admin image add

Admin Image Add umožňuje administrátorovi přidávat fotografie do systému. Umožňuje jak přidávání, odebírání, tak i změnu fotografií. Komponent přijímá nastavení pro hlavní (main), přejetí (hover), zobrazení modelu produktu (product model showcase) a detailní zobrazení (detail showcase). Tyto nastavení určují, které části komponentů se zobrazují, a které je tím pádem potřeba vyplnit.



Obr. 22 Ukázka komponentů Admin image add

Zdroj: Vytvořeno autorem bakalářské práce

3.4.2 Jednotlivé stránky

Aplikace je postavena na principu SPA (Single Page Application), jedná se o postup, při kterém aplikace zobrazuje obsah pouze na jedné stránce, která mění svůj obsah podle požadavků. Jednotlivé stránky byly navrženy tak, aby byly co nejvíce samostatné. To umožňuje příjemnou manipulaci s každou stránkou a její potenciální změnu. Nevýhodou tohoto přístupu je větší počet stránek a tedy čas potřebný na vývoj. Pro následující část jsem zvolil několik stránek aplikace pro podrobněší popis. (20)

Tabulka 4 Seznam stránek v aplikaci

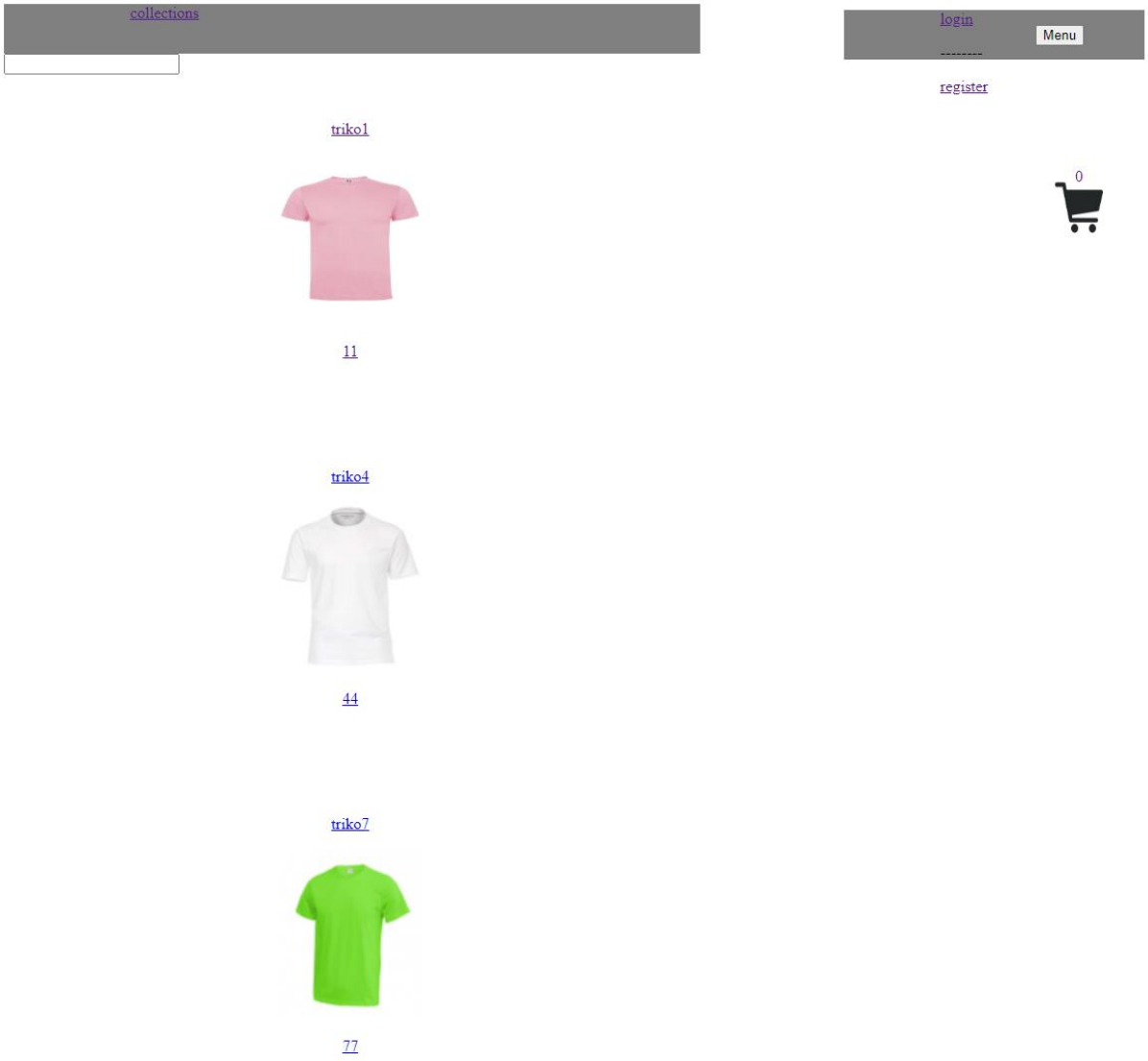
Název	Popis
Admin_page	Menu administrátora
Admin_refunds	Rozhraní pro manipulaci s reklamacemi
Admin_product_add	Rozhraní pro přidávání produktu
Admin_product_edit	Rozhraní pro editaci produktu
Admin_product_page	Rozhraní pro manipulaci s produkty
Admin_order_page	Rozhraní pro manipulaci s objednávkami
Admin_collction_add	Rozhraní pro přidávání kolekcí
Admin_collection_edit	Rozhraní pro editaci kolekcí
Admin_collection_page	Rozhraní pro manipulaci s kolekcemi
Showcase	Ukazuje produkty u vybrané kolekce
Register	Rozhraní pro registraci
Refund_log_of	Umožňuje odhlášenému uživateli podat reklamaci
Prepare_order	Stránka pro vytvoření objednávky
Order_complete	Stránka oznamující dokončení objednávky
Main	Hlavní stránka
Login	Rozhraní pro přihlášení
Item_info	Podrobné informace o produktu
Collections	Ukazuje kolekce, které k sobě mají produkty
Access_denied	Neoprávněný přístup
About	Ukazuje informace o firmě
User_menu	Menu uživatele
Orders	Ukazuje vytvořené objednávky

Add_delivery_info	Umožňuje přidat informace o doručení
Edit_delivery_info	Umožňuje editovat informace o doručení
Account_info	Ukazuje informace o účtu uživatele
Refunds	Ukazuje objednávky, které je možné reklamovat
Placed_returns	Ukazuje podané reklamace
Order_refund	Stránka na podání reklamace
Psw_change	Stránka na změnu hesla
New_psw	Stránka pro změnu hesla přihlášeného uživatele
Forgot_psw	Stránka na změnu hesla v případě ztráty
Code_psw	Stránka na ověření totožnosti uživatele
Stripe platebni brana	Stránka umožňuje uživateli provádět platby

Zdroj: Vytvořeno autorem bakalářské práce

Main page

Na hlavní stránce se nacházejí vybrané produkty, které chceme, aby uživatel první viděl. Může se jednat o nově přidané produkty nebo o produkty, kterých je potřeba se zbavit. Uživatel může tyto produkty filtrovat pomocí jejich názvu. Stránka obsahuje záhlaví, ve kterém jsou komponenty login hub pro přihlášení nebo registraci, menu pro prokliknutí na další stránky, košík oznamující počet produktů, které si uživatel plánuje zakoupit, a odkaz na kolekce, které se nacházejí v systému.



Obr. 23 Ukázka hlavní stránky

Zdroj: Vytvořeno autorem bakalářské práce

Collections page

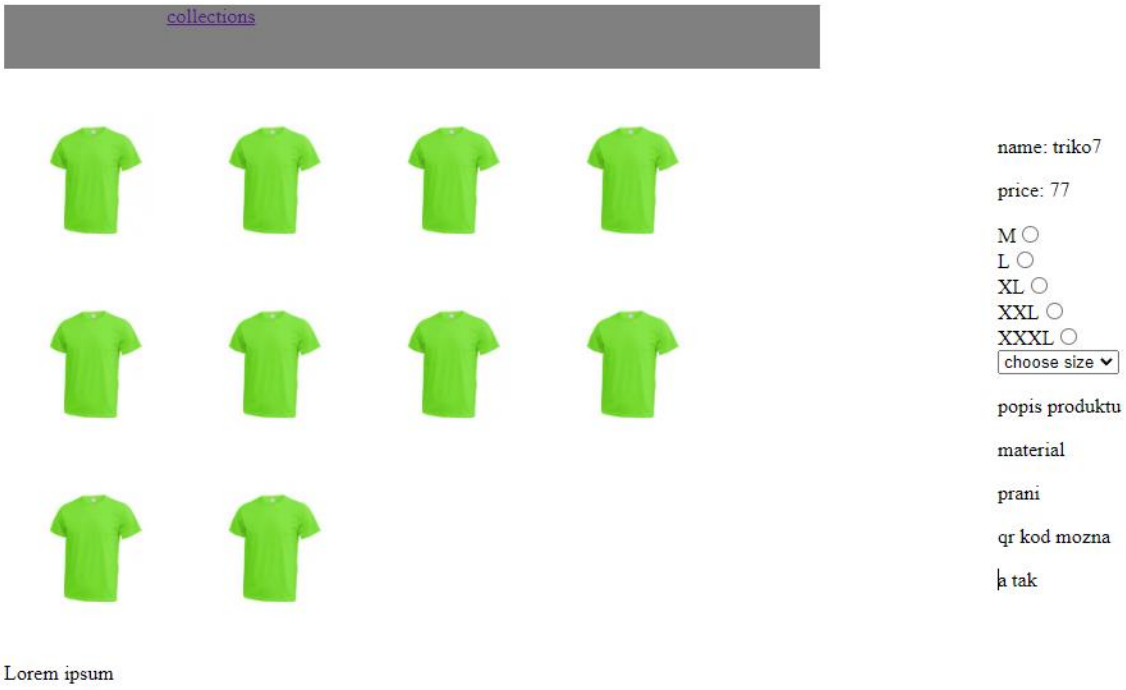
Zobrazuje kolekce, které jsou zadány v systému a jsou k nim přiřazeny určité produkty. Po kliknutí na konkrétní kolekci jsme přesměrováni na stránku, která nám ukazuje, jaké produkty jsou v dané kolekci.

Obr. 24 Ukázka stránky kolekcí

Zdroj: Vytvořeno autorem bakalářské práce

Item info page

Ukazuje podrobné informace o produktu, jako jsou název, cena, dostupné velikosti, popis produktu a fotografie produktu. Po vybrání vhodné velikosti a množství uživatel dostane možnost produkt buď přidat do košíku, nebo ho může rovnou zakoupit.

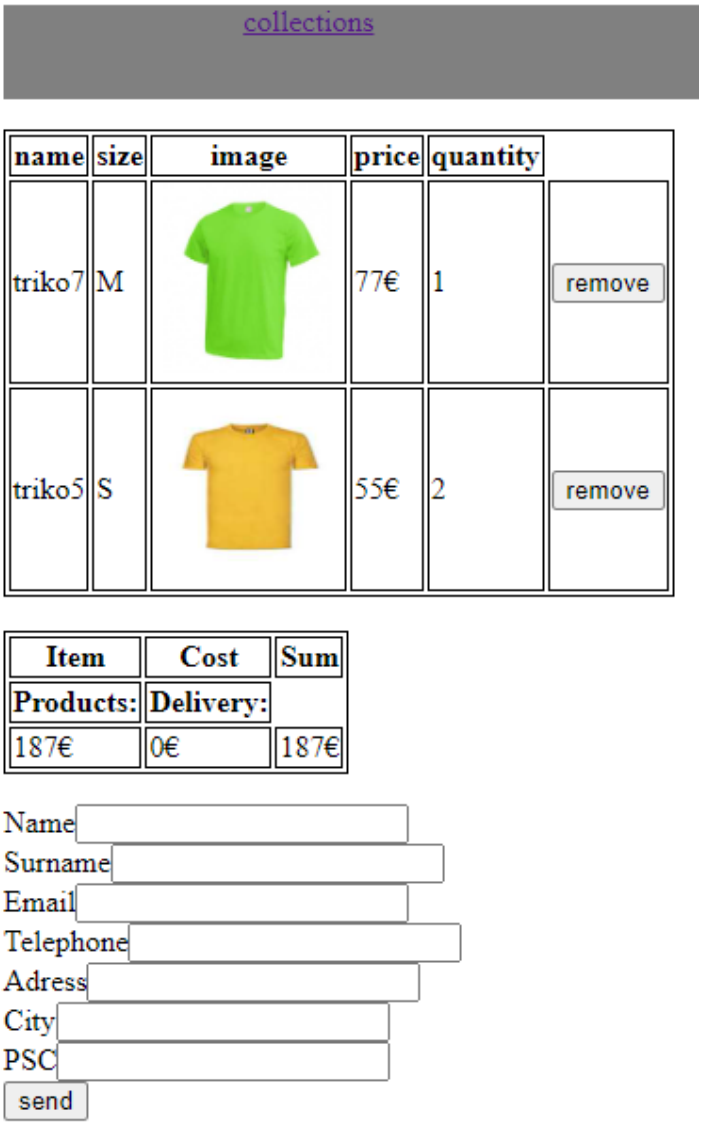


Obr. 25 Ukázka stránky s detaily produktu

Zdroj: Vytvořeno autorem bakalářské práce

Prepare order page

Finální stránka před nákupem ukazuje vše, co plánujeme zakoupit pomocí komponentů "Cart item & Money sub". Před nákupem je potřeba vyplnit formulář s údaji pro doručení, konkrétně jméno, příjmení, email, telefon, adresu, město a poštovní směrovací číslo. Pokud je uživatel přihlášený, nemusí vyplňovat údaje, stačí kliknout na jednu z uložených doručovacích adres a formulář se sám vyplní. Pokud nejsou údaje vyplněny, aplikace vypíše chybovou hlášku.

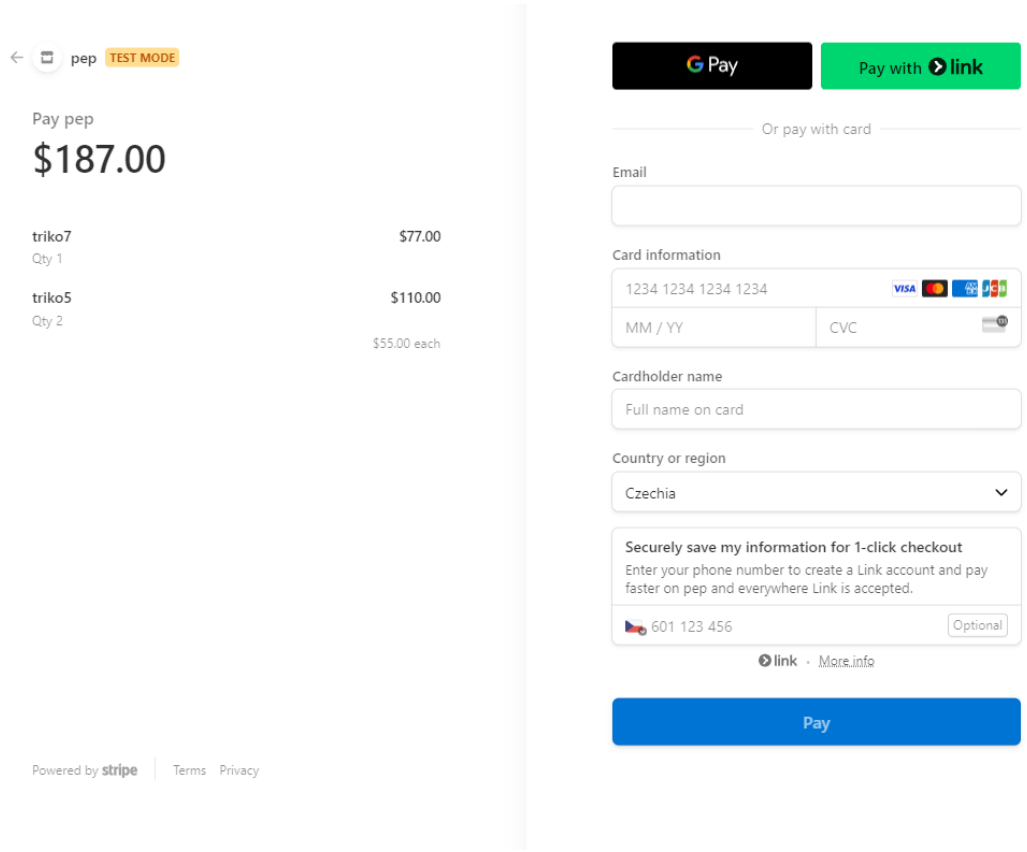


Obr. 26 Ukázka stránky pro shrnutí objednávky

Zdroj: Vytvořeno autorem bakalářské práce

Stripe platebni brana

Platební brána poskytnutá společností Stripe umožňuje uživateli finálně zakoupit vybrané produkty. Před platbou může uživatel zkontrolovat název a cenu produktu, které chce zakoupit. Po úspěšné platbě je objednávka uložena do systému a uživateli je zaslán email s účtenkou.

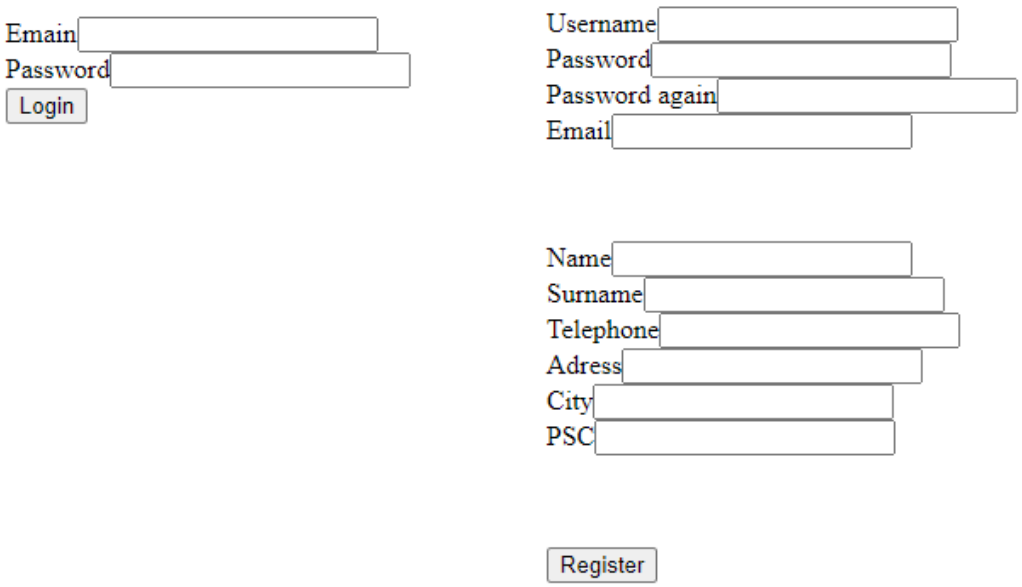


Obr. 27 Ukázka platební brány Stripe

Zdroj: API společnosti Stripe

Login page

Rozhraní pro registraci a přihlašování uživatelů umožňuje uživateli vytvořit nový účet nebo se přihlásit do stávajícího. Aplikace před registrací nebo přihlášením kontroluje data uživatele, jestli se nacházejí v systému, a následně vyhodnotí, jestli požadavku vyhoví, nebo zahlásí chybové hlášení.



Obr. 28 Ukázka stránky pro registraci a přihlášení

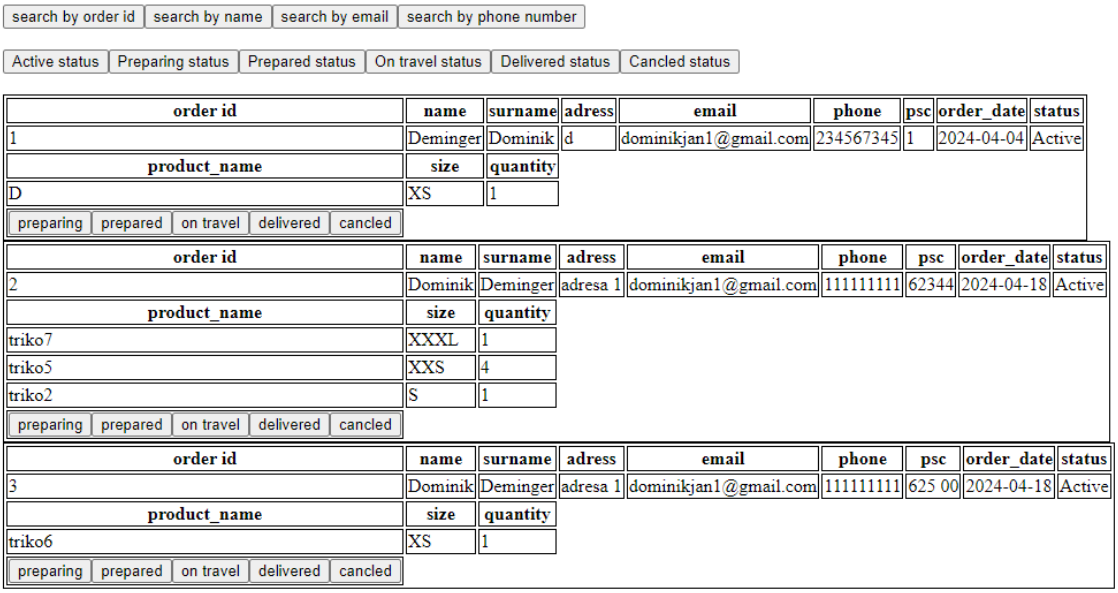
Zdroj: Vytvořeno autorem bakalářské práce

3.5.3 Rozhraní aministrátora

Následující část se věnuje popisu jednotlivých stránek obsažených v administrátorském rozhraní. Tyto stránky umožňují administrátorovi přístup k záznamům uloženým v databázi, a možnost s nimi manipulovat.

Refunds & Orders page

Umožňuje administrátorovi procházet a manipulovat s objednávkami a reklamacemi. Administrátor může záznamy třídit pomocí tlačítek v horní části stránky a přesně je může třídit podle identifikačního čísla, jména zákazníka, emailu zákazníka, telefonního čísla nebo podle statusu. Administrátor může také změnit status záznamu a tím oznámit zákazníkovi doručení nebo například zrušení objednávky.



Obr. 29 Ukázka rohraní administrátora pro manipulaci s reklamacemi a objednávkami

Zdroj: Vytvořeno autorem bakalářské práce

Admin menu

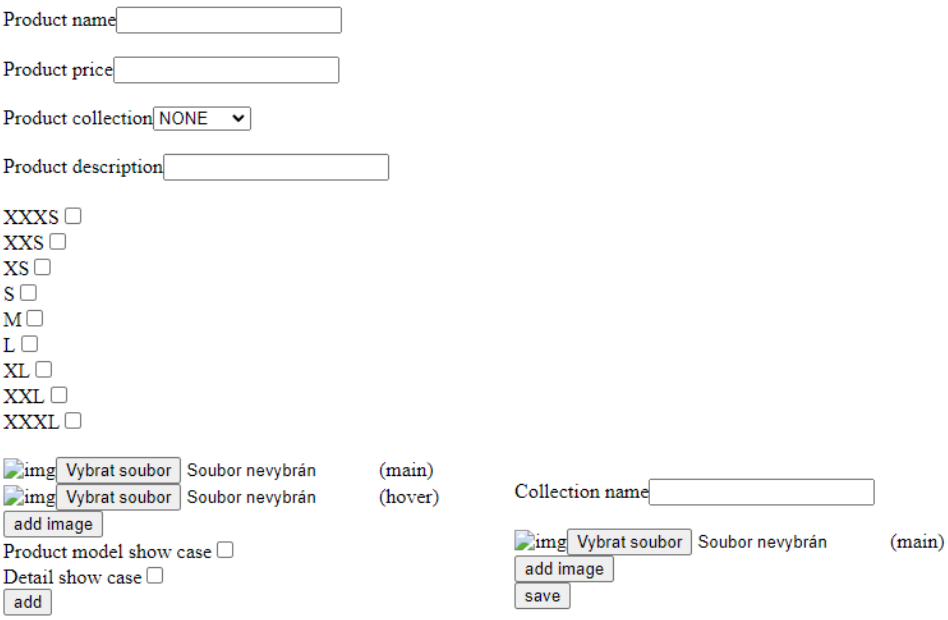
Menu administrátora umožňuje přístup ke stránkám pro správu záznamů v systému. Jedná se o stránky pro úpravu a přidávání kolekcí a produktů a manipulaci s reklamacemi a objednávkami.

Obr. 30 Ukázka menu administrátora

Zdroj: Vytvořeno autorem bakalářské práce

Přidání produktů & Kolekce

Administrátor na těchto stránkách může přidávat různé kolekce a produkty do systému. Před přidáním záznamu aplikace kontroluje, zda se záznam již v systému nenachází. Pokud ano, je vypsána chybová hláška. Aby bylo možné záznam přidat, je potřeba vyplnit formulář, přesněji data jako jsou název, cena, popis, fotografie atd.

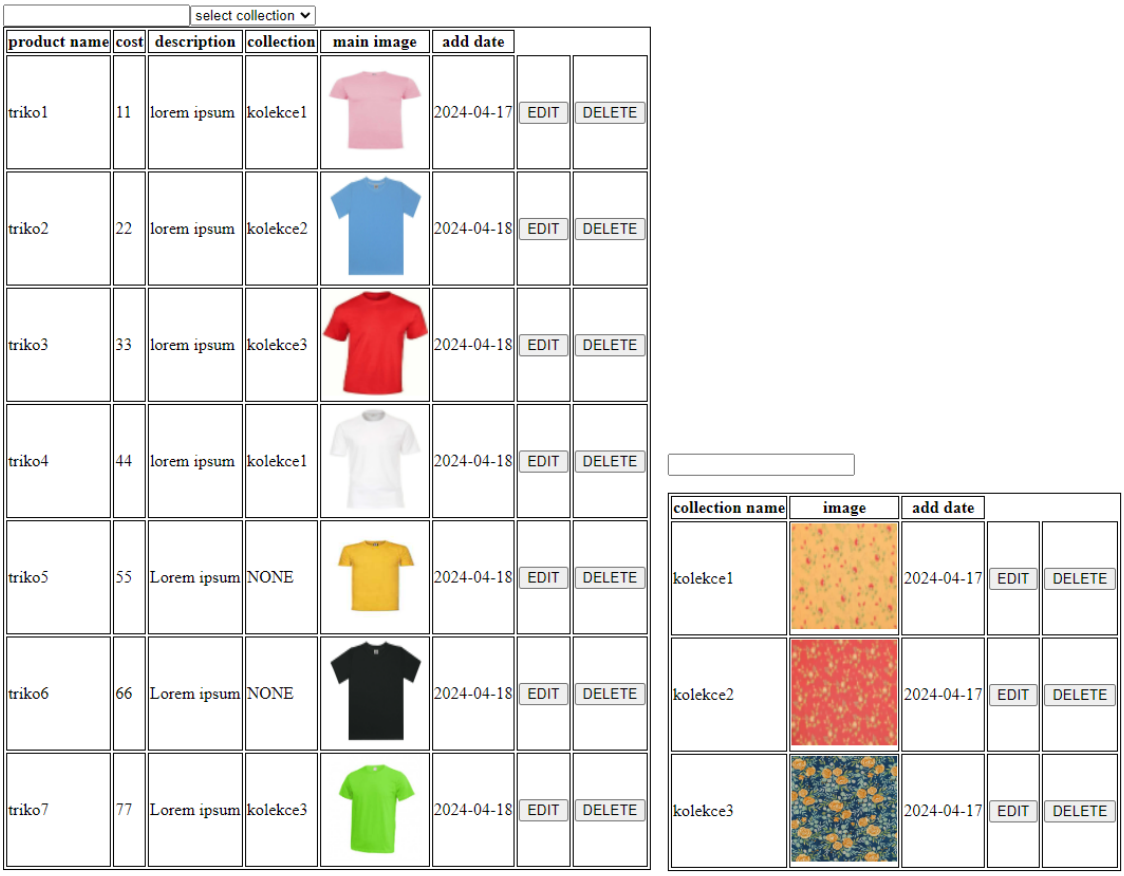


Obr. 31 Ukázka rozhraní administrátora pro přidávání produktů a kolekcí

Zdroj: Vytvořeno autorem bakalářské práce

Collection & Product page

Tyto stránky zobrazují kolekce a produkty v systému. Administrátor má možnost smazat určité záznamy nebo se může prokliknout na stránku, která mu umožní záznamy editovat.

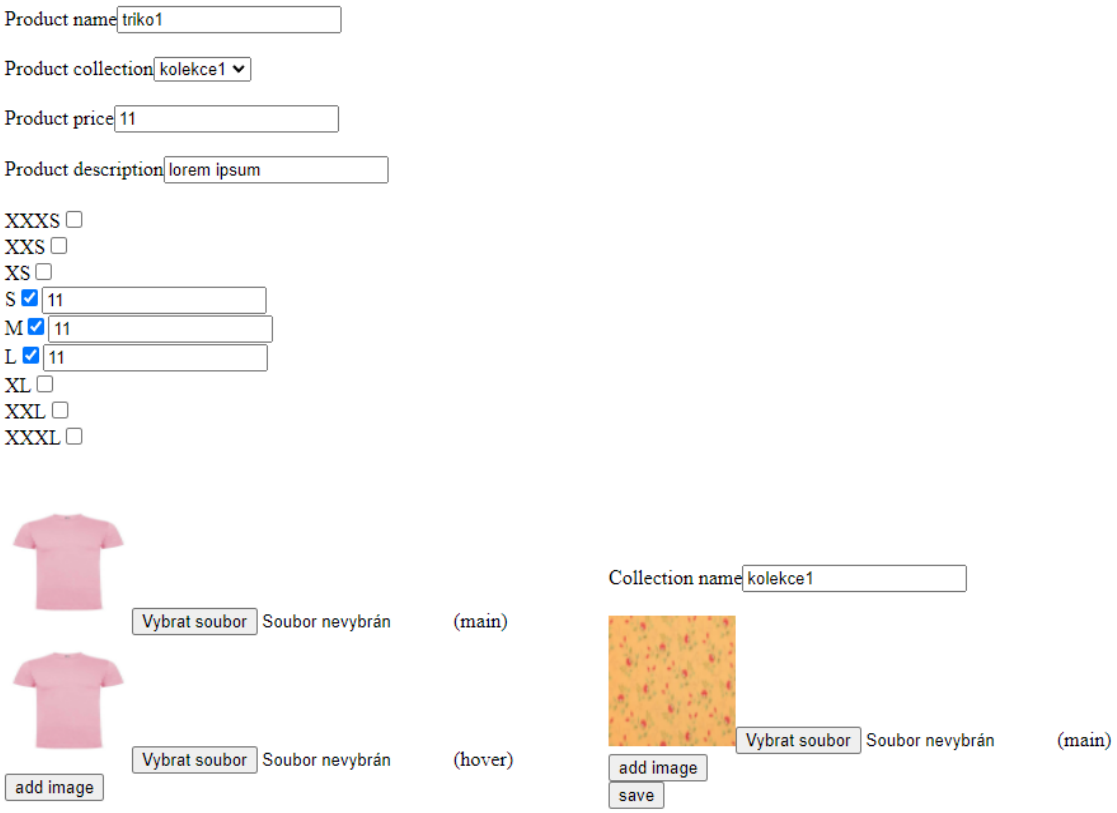


Obr. 32 Ukázka rozhraní administrátora pro manipulaci s kolekcemi a produkty

Zdroj: Vytvořeno autorem bakalářské práce

Edit collection & Edit product

Stránky umožňují editovat záznamy v databázi. Před změnou záznamu systém provede kontrolu, jestli se upravovaný záznam nenachází v databázi. Pokud ano, vyhodí chybovou zprávu, pokud ne, záznam je uložen. S záznamem v databázi jsou aktualizovány i fotografie na serveru.



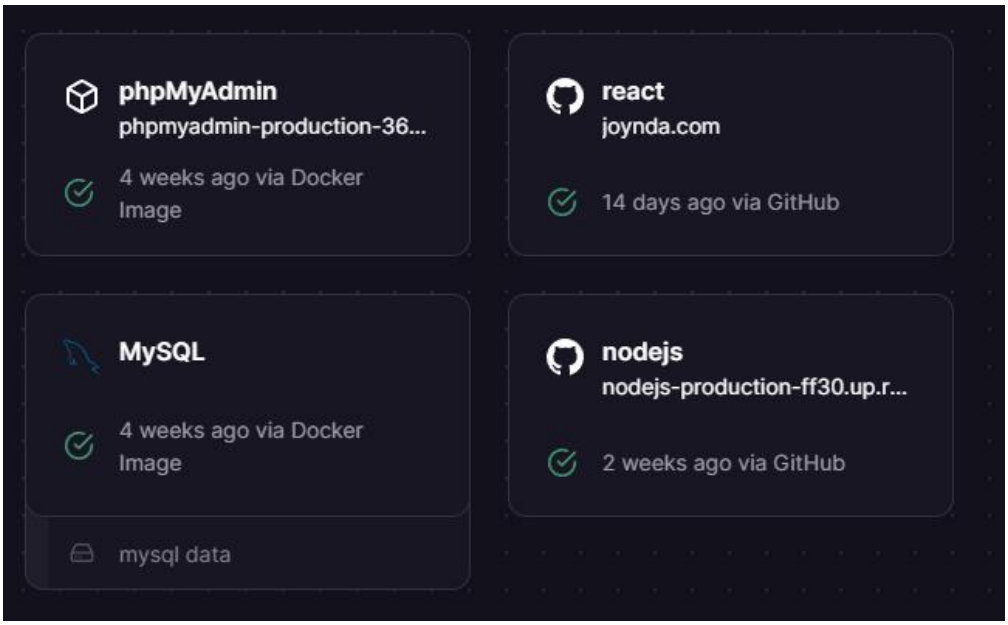
Obr. 33 Ukázka rozhraní administrátora pro editaci kolekcí a produktů

Zdroj: Vytvořeno autorem bakalářské práce

3.5 Implementace

Aby aplikace mohla plnit svůj požadovaný záměr, musí být dostupná pro uživatele, kteří ji chtějí použít. Na toto slouží hostovací platformy jako je Heroku nebo Render, které poskytují prostředky, jež umožňují aplikaci dostat se do online světa. Existuje velké množství poskytovatelů těchto služeb, a rozdíly mezi nimi mohou být v cenách služeb i ve službách samotných. Některé platformy poskytují pouze hostování serverové části aplikace, nebo naopak pouze přední (frontendové) části.

Pro uvedení aplikace do chodu jsem zvolil Railway. Platforma Railway poskytuje široké množství možností hostování jak databází, jako jsou například MySQL, MongoDB a PostgreSQL, tak i ostatních nástrojů jako jsou Flask, TypeBot nebo WordPress. Railway jsem zvolil díky jednoduché manipulaci a nasazení aplikace a také proto, že umožňuje mít všechny části aplikace na jednom místě a usnadňuje tak jejich správu. Railway poskytuje možnost za poplatek nasadit jak Node.js serveru, tak MySQL databázi a samozřejmě i přední části aplikace zastoupenou Reactem. Velkou výhodou je možnost použít rozhraní phpMyAdmin, které zjednodušuje manipulaci s databází a umožňuje nahrát schéma databáze pro rychlejší a pohodlnější implementaci. (12)



Obr. 34 Ukázka rozhraní Railway aplikace

Zdroj: Aplikace Railway

Implementace pomocí GitHubu

Velkou výhodou Railway je její kompatibilita s GitHubem. GitHub je platforma široce rozšířená mezi vývojáři. Umožňuje ukládat jednotlivé verze aplikace přímo v aplikaci, což umožňuje jednoduchou manipulaci a možnou aktualizaci projektu. GitHub na oficiálních stránkách nabízí zdarma ke stažení instaler, který nám umožní používat Git příkazy přímo v naší aplikaci. Po nainstalování můžeme s GitHubem interagovat pomocí konzole a nahrát náš projekt do úložiště pomocí git příkazů. (11)

- git init - Převede libovolný adresář na Git repozitář
- git add – Přidá všechny nové nebo upravené soubory do inspekčního prostoru, kde jsou připraveny na nahrání do úložiště
- git commit - Vytváří commit, což je jako snímek vašeho repozitáře
- git push – Nahrává všechny lokální commity do repozitáře.

Po každé změně v Git repozitáři je Railway upozorněna a aktualizuje jednotlivé komponenty, které byly změněny. (11)

Závěr

Cílem práce bylo vytvořit funkční prototyp aplikace pro společnost Joynda s.r.o. na základě požadavků majitele. Pro vývoj aplikace bylo potřeba zvolit vhodné technologie a postupy, které umožní implementovat všechny požadavky a umožní aplikaci další vývoj a rozšíření. Pro tvorbu aplikace byla využita knihovna React pro přední část aplikace, která napomáhá ve vývoji Single Page Applications (SPA) a usnadňuje manipulaci s kódem díky prvkům jako jsou komponenty nebo React Hooks. Serverová část (backend) byla vytvořena pomocí běhového prostředí (runtime environment) Node.js s podporou frameworku Express. Aplikace potřebuje uložistě, kde může skladovat záznamy pro naslednou manipulaci. Tento problém byl vyřešen implementací databáze, přesněji MySQL database, do aplikace.

Hlavním požadavkem na aplikaci bylo rozhraní pro administrátora, kterému umožní jednoduše manipulovat s aplikací, a rozhraní pro uživatele, kterému umožní nákup požadovaných produktů a také ukládání důležitých informací přímo do aplikace. Požadavky byly statoveny majitelem firmy a jeho vize co se týče funkcianality a implentace byly naplněny.

V této chvíli se jedná o prototyp aplikace. Cílem prototypu bylo implementovat požadovanou funkcionalitu aplikace tak, aby ji bylo možné použít pro obchodní závazky firmy. Prototyp neobsahuje žádné grafické úpravy, firma má vlastní grafické designéry, kterým bude aplikace předána a přetvořena podle jejich vize.

Firma má v plánu růst, a s tím i rozšiřovat aplikaci. V tuto chvíli jsou v aplikaci pouze nezbytné funkce, které jí umožňují chod. Do budoucnosti firma plánuje rozšíření aplikace, co se týče funkcionality, jako například rozšíření vyhledávání produktů v systému, implementace systému pro generaci a tisk formulářů pro odesílání reklamací nebo optimalizace systému.

V závěru hodnotím zkušenost s projektem a jeho vývojem velmi kladně. Práce se zadavatelem byla velmi příjemná a přínosná, projekt splňuje všechny cíle a očekávání a je připraven na další rozšíření.

Seznam použité literatury

(1) TechTarget Contributor. *web application (web app)*. Online. techtarget.com. Poslední aktualizace: leden 2023. Dospuné z: https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app [cit. 25.4. 2024]

(2) SHARMA, Neeraj, Liviu PERNIU, Raul F. CHONG, Abhishek IYER, Chaitali NANDAN, Adi-Cristina MITEA, Mallarswami NONVINKERE, Mirela DANUBIANU. *Database Fundamentals*. First Edition. Online. 2010. Dostupné z: https://my.uopeople.edu/pluginfile.php/57436/mod_book/chapter/37615/CS2203.Textbook.Database.Fundamentals.pdf [cit. 25.4. 2024]

(3) MongoDB. *What is NoSQL?*. Online. mongodb.com. Dostupné z: https://www.mongodb.com/resources/basics/databases/nosql-explained [cit. 5.5. 2024]

(4) DB-Engines. *DB-Engines Ranking*. Online. db-engines.com. Dostupné z: https://db-engines.com/en/ranking [cit. 5.5. 2024]

(5) Oracle. *What is MySQL?*. Online. oracle.com. Dostupné z: https://www.oracle.com/mysql/what-is-mysql/ [cit. 25.4. 2024]

(6) WEINBERG, Paul, James GROFF, Andrew OPPEL. *SQL The Complete Reference*. Third Edition. Online. McGraw-Hill, 2010. ISBN: 978-0-07-159256-7. Dostupné z: https://ci-ceit.edu.ck/wp-content/uploads/2021/01/sql-the-complete-reference-third-edition-sep-2009.pdf [cit. 5.5. 2024]

(7) HAVERBEKE, Marijn. *Eloquent JavaScript*. Fourth Edition. Online. 2024. Dostupné z: https://eloquentjavascript.net/00_intro.html [cit. 1.5. 2024]

(8) Typescriptlang. *The TypeScript Handbook*. Online. typescriptlang.org. Dostupné z: https://www.typescriptlang.org/docs/handbook/intro.html [cit. 1.5. 2024]

(9) QURESHI, Ashad ullah. *React JS Notes for Professionals*. Online. Concepts Books Publication, 2023. ISBN: 9798386385798. Dostupné z: https://books.google.cz/books/about/React_JavaScript_Notes_For_Professionals.html?id=s9eyEAAAQBAJ&redir_esc=y [cit. 25.5. 2024]

(10) MEAD, Andrew. *Learning Node.js Development*. Online. Packt Publishing Ltd., 2018. ISBN: 978-1-78839-554-0. Dostupné z: https://edu.anarcho-copy.org/Programming%20Languages/Node/learning-nodejs-development.pdf [cit. 25.5. 2024]

(11) GitHub. *Oficiální dokumentace GitHub*. Online. docs.github.com. Dostupné z: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git [cit. 1.5. 2024]

(12) Railway. *Oficiální dokumentace Railway*. Online. docs.railway.app. Dostupné z: https://docs.railway.app/overview/about-railway [cit. 1.5. 2024]

(13) Stripe. *Oficiální dokumentace Stripe*. Online. docs.stripe.com. Dostupné z: https://docs.stripe.com [cit. 1.5. 2024]

(14) RUMBAUGH, James, Ivar JACOBSON, Grady BOOCH. *The Unified Modeling language Reference Manual*. Second Edition. Online. Addison-Wesley Professional, 2004. ISBN: 0-321-24562-8. Dostupné z: https://personal.utdallas.edu/~chung/Fujitsu/UML_2.0/Rumbaugh--UML_2.0_Reference_CD.pdf [cit. 1.5. 2024]

(15) BAGUI, Sikha Saha, Richard Walsh EARP. *Database Design Using Entity-Relationship Diagrams*. Third Edition. Online. CRC Press, 2023. ISBN: 978-1-032-01718-1. Dostupné z: https://download.bibis.ir/Books/Database/2022/Database-Design-Using-Entity-Relationship-Diagrams-by-Sikha-Saha-Bagui-Richard-Walsh-Earp_bibis.ir.pdf [cit. 1.5. 2024]

(16) NECULA, Sabina. *Exploring the Model-View-Controller (MVC) Architecture: A Broad Analysis of Market and Technological Applications*. Online. 2024.

Dostupné z: https://www.researchgate.net/publication/380197155_Exploring_The_Model-View-Controller_MVC_Architecture_A_Broad_Analysis_of_Market_and_Technological_Applications [cit. 1.5. 2024]

(17) BROWN, Ethan. *Web Development with Node and Express*. First Edition. Online. O'Reilly Media, Inc., 2014. ISBN: 978-1-491-94930-6. Dostupné z: https://www.vanmeegern.de/fileadmin/user_upload/PDF/Web_Development_with_Node_Express.pdf [cit. 1.5. 2024]

(18) Redhat. *What is a webhook?*. Online. redhat.com. Poslední aktualizace: 1.2. 2024. Dostupné z: https://www.redhat.com/en/topics/automation/what-is-a-webhook [cit. 5.5. 2024]

(19) CLARKE, Justin. *SQL Injection Attacks and Defence*. Second Edition. Online. Syngress, 2012. ISBN: 978-1-59749-963-7. Dostupné z: https://books.google.cz/books?hl=cs&lr=&id=Spm7UgBwzjIC&oi=fnd&pg=PR3&dq=sql+injection&ots=k2-BYLdgeE&sig=maDmHzkT7deVWg4itYtOLQtNI_w&redir_esc=y#v=onepage&q=sql%20injection&f=false [cit. 5.5. 2024]

(20) FLATOW, Ido, Gil FINK. *Pro Single Page Application Development: Using Backbone.js and ASP.NET*. Online. Apress, 2014. ISBN: 978-1-4302-6674-7. Dostupné z: https://pepa.holla.cz/wp-content/uploads/2015/10/Pro-Single-Page-Application-Development.pdf [cit. 5.5. 2024]

Seznam obrázků

Seznam tabulek