UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2024 Siarhei Semakou

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Tvorba eshopu s elektrotechnikou a systémem elektronického hodnocení spolehlivosti

Bakalářská práce

2024 Siarhei Semakou

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

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

Jméno a příjmení:	**Siarhei Semakou**
Osobní číslo:	**I20153**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Tvorba eshopu s elektrotechnikou a systémem elektronického hodnocení spolehlivosti**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Projekt bude obsahovat funkci hodnocení spolehlivosti, kdy zákazník bude mít možnost hodnotit a psát recenze o produktech. Dále v projektu bude možnost získávat kurzy různých měn z veřejné bankovní API a možnost vybrat měnu pro nákup. Pro uchování a správu dat je plánovano využít databázi Oracle s přístupem prostřednictvím JDBC. Backend aplikace bude postaven na Spring Boot.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

WALLS, Craig. Spring in Action. Sebastopol: O'Reilly Media, 2018. ISBN 978-1-61729-494-5.
CHINNATHAMBI, Kirupa. Learning React. Boston: Addison-Wesley Professional, 2016. ISBN 978-0--13454-631-5.
KLEPPMANN, Martin. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. Sebastopol: O'Reilly Media, 2017. ISBN 978-1-44937-332-0.

Vedoucí bakalářské práce: **Ing. Jan Panuš, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2024

Prohlašuji:

Práci s názvem Tvorba eshopu s elektrotechnikou a systémem elektronického hodnocení spolehlivosti jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

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

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

V Pardubicích dne 28. 02. 2024

Siarhei Semakou v. r.

PODĚKOVÁNÍ

Chtěl bych poděkovat mým rodičům za jejich podporu během mého studia.

ANOTACE

Úkolem tohoto projektu je vytvořit webovou aplikaci elektronického obchodu. Aplikace bude poskytovat uživateli funkce jako registrace a přihlašování, přístup k aplikaci jako host, filtrování produktů podle různých kritérií, přidávání produktů do košíku, provádění nákupů a správa dat podle přístupových práv. V teoretické části bude provedeno zkoumání, které problémy a jakým způsobem ty problémy podobné projekty řeší. V praktické části bude popsáno, jakými technologiemi byl projekt vytvořen a jaké přístupy byly použity. Při vytváření aplikace byly použity programovací jazyky Java a JavaScript, frameworky Spring a React a databáze Oracle.

KLÍČOVÁ SLOVA

Eshop, webová aplikace, Java, JavaScript, Spring framework, React.

TITLE

Creating an eshop with electrical equipment and a system for electronic reliability assessment.

ANNOTATION

The task of this project is to create a web application for an electronic commerce platform. The application will provide users with functions such as registration and login, guest access to the application, product filtering based on various criteria, adding products to the cart, making purchases, and data management based on access rights. In the theoretical part, research will be conducted on which problems similar projects face and how they address them. The practical part will describe the technologies used in the project and the approaches taken. Programming languages Java and JavaScript, frameworks Spring and React, and Oracle database were used in the development of the application.

KEYWORDS

Eshop, web application, Java, JavaScript, Spring framework, React.

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

REST	Representational State Transfer
API	Application Programming Interface
SPA	Single Page Application
RDMBS	Relational Database Management System
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
JSON	JavaScript Object Notation
GUI	Graphical User Interface
XML	Extensible Markup Language

ÚVOD

V moderním světě často člověku chybí čas a někdy i fyzické síly k návštěvě obchodu osobně. Nejjednodušším řešením je použití internetového obchodu a vybrání potřebných produktů. V současné době existuje několik obrovských internetových obchodů, jako je Amazon, Ebay, Aliexpress. Existuje také velké množství internetových obchodů se specifickým druhem zboží, jako jsou elektronické obchody, oděvní obchody, sportovní obchody atd. S nástupem internetu získává vývoj webových aplikací stále větší popularitu a více firem potřebuje webové stránky, na kterých může uživatel snadno a rychle nakupovat zboží.

Cílem bakalářské práce je vytvořit webovou aplikaci elektronického obchodu, která bude poskytovat uživateli funkce jako přístup k aplikaci podle oprávnění, filtrování produktů podle různých kritérií, přidávání produktů do košíku, provádění nákupů a správa dat. Úspěšný vývoj takového obchodu vyžaduje pečlivé plánování a stanovení konkrétních cílů, které by měly být dosaženy. Celý projekt je zaměřen na jednoduchost použití a byl založen na aktuálních řešeních na globálním trhu. Hlavním nápadem bylo vytvořit základ pro snadné rozšiřování a rozvoj aplikace pro internetový obchod elektroniky.

Klientská část projektu (frontend) byla řešená pomocí jazyka JavaScript a webového frameworku React. Serverová část projektu (backend) byla řešená pomocí jazyka Java a frameworku Spring Boot. Backend komunikuje s databází Oracle, která je umístěna na lokálním serveru.

V teoretické části se nachází zkoumání problematiky elektronických obchodů a identifikace klíčových faktorů úspěchu eshopu. Praktická část práce se zaměřuje na konkrétní implementaci navrženého eshopu. Nejprve je představen celkový návrh systému, který zahrnuje architekturu aplikace, rozložení funkcí a identifikaci hlavních komponent. Stručně je popsána funkcionalita aplikace a role jednotlivých uživatelů v systému. V následující kapitole "Použité nástroje" je zaměřeno na softwarové nástroje, které byly použity při vývoji webové aplikace. Jsou popsány jejich funkce, výhody a způsoby integrace do vývojového procesu. Kapitola "Použité technologie" se zaměřuje na technologie, jako jsou programovací jazyky, frameworky a knihovny, které byly využity při implementaci aplikace. V další kapitole "Databáze" je detailně popsán datový model aplikace a struktura databáze. Jsou zahrnuty informace o entitách a vztazích mezi nimi. Na závěr je představena ukázka vyvinuté aplikace a zdrojových kódů v kapitole "Demonstrace aplikace".

1 ANALÝZA PROSTŘEDÍ

1.1 Koncept internetového obchodu

Internetový obchod, známý také jako eshop či online obchod, je elektronická platforma umožňující firmám a jednotlivcům prodávat své produkty a služby prostřednictvím internetu. Hlavními charakteristikami internetového obchodu jsou možnost prohlížení a nákupu zboží online, snadná platba elektronickými metodami a rychlé dodání zakoupeného zboží zákazníkovi. Internetový obchod může nabízet široký sortiment produktů od elektroniky až po oblečení nebo potraviny a může fungovat jak samostatně nebo jako doplněk k tradičnímu obchodu.

1.2 Cíle a úkoly vývoje internetového obchodu

Zavedení internetového obchodu má následující cíle:

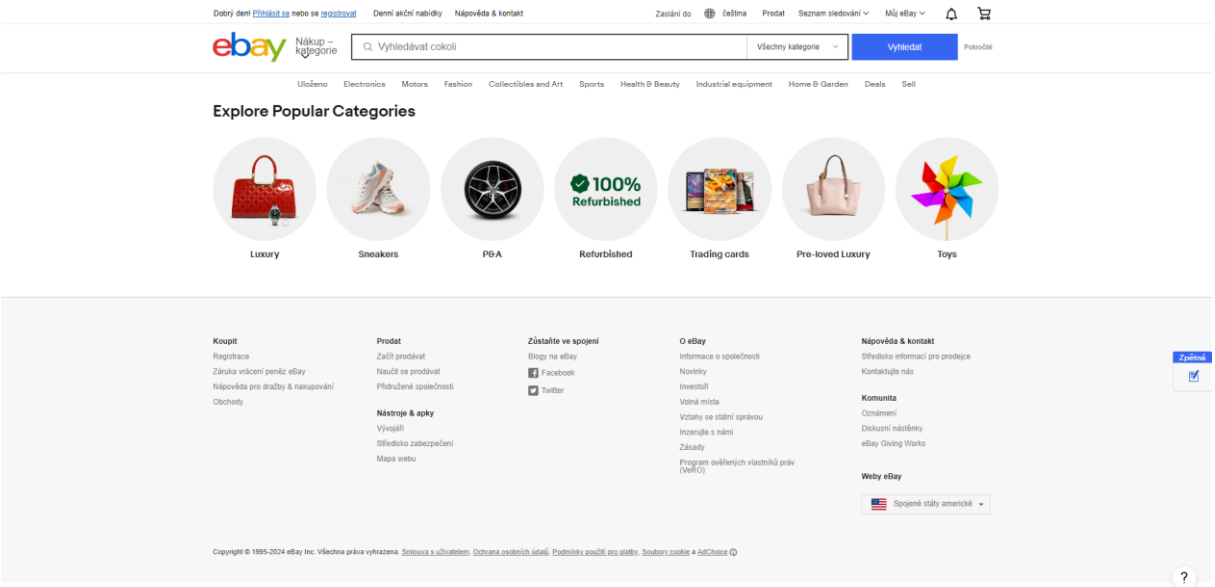
- Zvýšení tržního podílu a získání nových zákazníků.
- Snížení pracovní náročnosti manažerů pro realizaci.
- Snížení nákladů na zásobení hlavních činností (prodeje).
- Snížení počtu chyb v účetnictví a zvýšení rychlosti její přípravy.

Vytvářený internetový obchod musí zaručit plnění všech běžných funkcí pro podobné projekty, a to konkrétně:

- Možnost registrace uživatelů.
- Možnost vytvoření objednávky z libovolného množství zboží.
- Existence víceúrovňových kategorií zboží s jednoduchým vyhledáváním podle zadaných kritérií.
- Osobní účet registrovaných uživatelů.
- Košík, kam si zákazníci přidávají vybrané produkty.
- Profil uživatele s zobrazením objednávek a osobní peněženka.
- Možnost psát komentáře a recenze o produktu.

1.3 Analýza podobných projektů

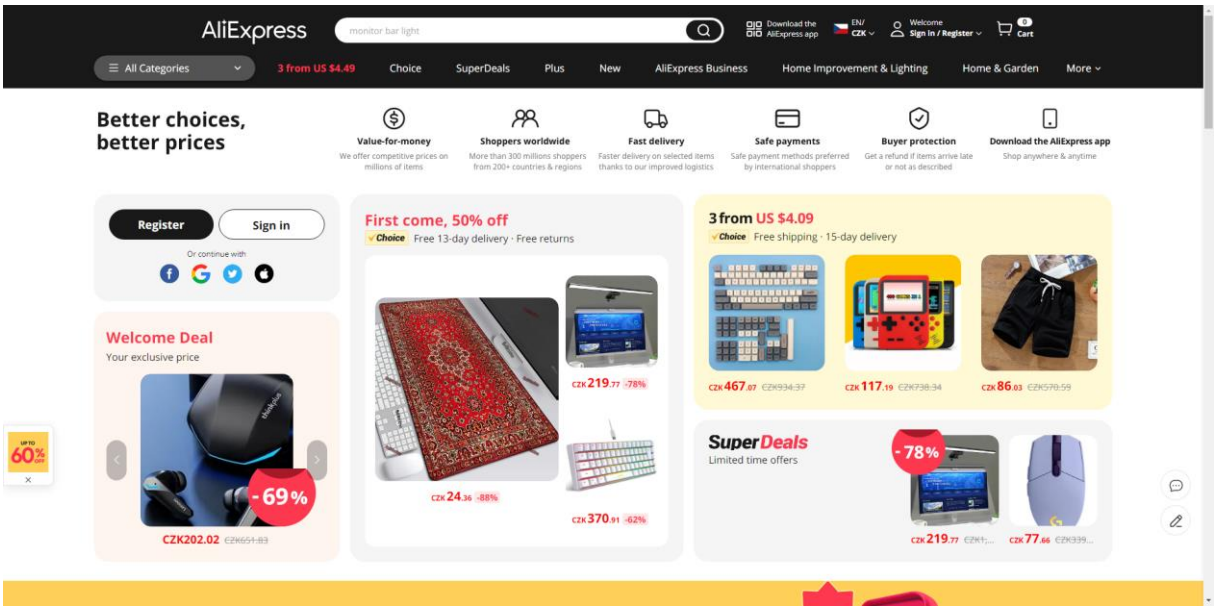
1.3.1 Ebay.com



Obrázek 1: *https://www.ebay.com/*

Tento internetový obchod se zaměřuje na prodej zboží a služeb prostřednictvím internetu. V tomto eshopu se prodávají produkty od lidí a firem z celého světa. Obsahuje následující sekce: hlavní stránka, kde jsou umístěny nové a doporučené produkty, navigace podle kategorií zboží, například elektronika, móda, krása a zdraví. Zde jsou také odkazy na jiné zdroje včetně online služeb. Dále na stránce najdete informace o webu, kontaktní údaje, platební metody a veškeré potřebné další informace pro zákazníky. V každém katalogu jsou k dispozici podkategorie pro snadnější navigaci a vyhledávání požadovaných produktů. Odkazy na registraci, vývojáře, mapu stránek a další jsou umístěny v dolní části. Ebay byl vytvořen 4. září 1995 v San Jose (Kalifornie) programátorem Pierrem Omidyarem.

1.3.2 Aliexpress.com



Obrázek 2: *https://www.aliexpress.com/*

AliExpress je jedním z největších a nejvýznamnějších internetových obchodů v Číně. Jeho hlavním cílem je prodej zboží a služeb. Na domovské stránce jsou zobrazeny hlavní nabídky obsahující oblíbené kategorie a doporučené produkty. Ve footeru stránky jsou umístěny důležité články a sekce pro zpětnou vazbu, kontaktní informace a mnoho dalších sekcí. Hlavním nástrojem pro vyhledávání a výběr produktů je navigační menu, kde si zákazník může snadno najít požadovanou kategorii produktu a samotný produkt. Na domovské stránce je také umístěn formulář pro přihlášení registrovaných zákazníků, po autorizaci bude k dispozici osobní účet a košík s vybranými produkty. Tento internetový obchod byl vyvinut Jackem Má, který založil skupinu Alibaba v roce 1999. Dnes má společnost vlastní elektronický platební systém Alipay, internetové obchody Taobao.com a AliExpress.

Oba tyto internetové obchody zahrnují různé aspekty elektronického obchodování, včetně prodeje různých produktů a služeb, pohodlných funkcí vyhledávání a navigace, možnost vytvoření osobního účtu pro uživatele a systém zpětné vazby a podpory. Studium principů fungování a funkcionality těchto velkých internetových obchodů pomáhá lépe porozumět požadavkům uživatelů a lépe navrhnout a vyvinout vlastní projekt.

2 NÁVRH SYSTÉMU

2.1 K čemu slouží

Tato aplikace je určena jak pro jednotlivé zákazníky, kteří chtějí pohodlně nakupovat elektroniku online, tak i pro správce obchodů, kteří hledají efektivní způsob pro provoz svého elektronického obchodu. Celkově je aplikace navržena tak, aby poskytovala uživatelům intuitivní a pohodlné prostředí pro nakupování elektroniky online a zároveň umožňovala správcům obchodů efektivní správu a rozvoj svých podniků. Pro zákazníka představuje pohodlnou a efektivní cestu, jak nakupovat elektroniku přímo z pohodlí domova. S rozsáhlým katalogem produktů, snadným vyhledáváním a rychlým procesem objednávání poskytuje tento projekt uživatelsky přívětivé prostředí, které usnadňuje nakupování a šetří čas. Pro majitele obchodu představuje tato webová stránka efektivní nástroj pro rozvoj jejich podnikání. Poskytuje jim platformu pro prezentaci a prodej jejich produktů širokému spektru zákazníků. Díky online přítomnosti mají obchodníci možnost oslovit a přilákat nové zákazníky a rozšířit svůj trh.

2.2 Role a přístup

Aplikace umožňuje tři hlavní role: anonymního uživatele (host), registrovaného uživatele a administrátora. Každá z těchto rolí má své specifické oprávnění a možnosti v systému.

2.2.1 Anonymní uživatel

Anonymní uživatel je neregistrovaný návštěvník aplikace. Jeho přístup k funkcím je omezen. Hlavními funkcemi, ke kterým má přístup, jsou:

- Procházení produktů: Anonymní uživatel může procházet nabídku produktů a zobrazovat jejich detaily, včetně cen a specifikací.
- Hledání: Má možnost vyhledávat produkty pomocí klíčových slov a kritérií.
- Registrace: Může se registrovat, aby získal další výhody.

2.2.2 Registrovaný uživatel

Registrovaný uživatel je ten, kdo se úspěšně zaregistroval v aplikaci a přihlásil se. Jeho úroveň přístupu je rozšířena ve srovnání s anonymním uživatelem. Jeho hlavními funkcemi jsou:

- Přihlášení: Registrovaný uživatel se může přihlásit do aplikace.

- Psaní komentářů a hodnocení: Registrovaný uživatel má možnost psát komentáře k produktům a udělovat jim hodnocení.
- Přidávání do košíku: Může přidávat produkty do svého košíku a spravovat je.
- Správa profilu: Může upravovat svůj profil.

2.2.3 Administrátor

Administrátor je uživatel s nejvyššími oprávněními v aplikaci. Jeho role je spravovat a řídit chod aplikace. Jeho funkce zahrnují:

- Správa produktů: Administrátor může přidávat, upravovat nebo mazat produkty z katalogu.
- Správa uživatelů: Má kontrolu nad uživatelů, jejich přístupovými právy a údaji.
- Správa objednávek: Může zobrazovat a spravovat objednávky.

2.3 Analýza požadavků

2.3.1 Funkční požadavky

Při přístupu na stránky internetového obchodu nezaregistrovaný uživatel okamžitě zobrazí hlavní stránku s doporučenými produkty a možností pro filtrování a vyhledávání zboží. Kliknutím na konkrétní produkt má uživatel možnost prohlédnout si podrobný popis, komentáře a hodnocení tohoto produktu. Pro nákup produktů musí uživatel nejprve provést registraci kliknutím na červené tlačítko v horní části stránky a poté přidat požadovaný produkt do košíku. Z košíku může uživatel nakoupit produkty, pokud má dostatek peněz na svém virtuálním účtu.

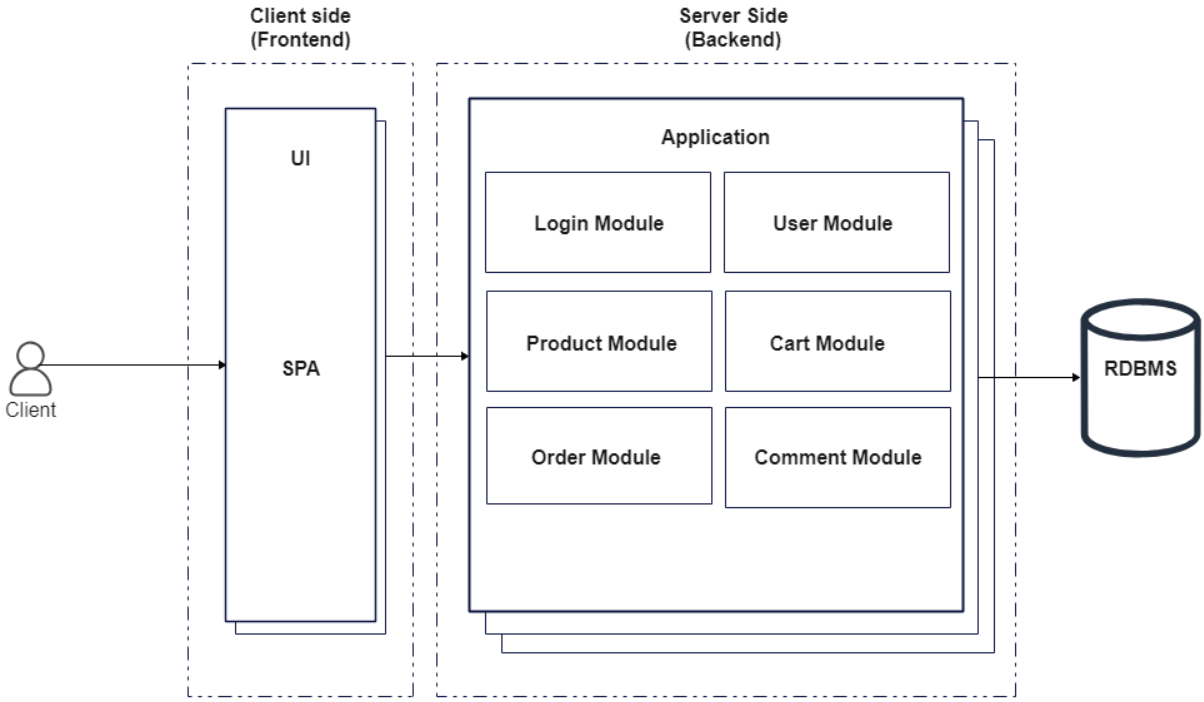
2.3.2 Nefunkční požadavky

Nefunkční požadavky pro eshop podobný Aliexpress nebo eBay zahrnují klíčové aspekty použitelnosti, efektivity, spolehlivosti, dodržování standardů a bezpečnosti. Použitelnost zajišťuje, že uživatelské rozhraní je intuitivní a snadno přístupné, což umožňuje uživatelům rychle a pohodlně nalézt a zakoupit produkty. Efektivita systému je nezbytná pro rychlé načítání stránek a minimalizaci doby odezvy i při vysoké návštěvnosti. Spolehlivost eshopu se projevuje v minimalizaci neplánovaných výpadků a schopnosti automatického zotavení, čímž se zajišťuje nepřetržitý provoz. Dodržování průmyslových standardů a legislativních požadavků, jako je GDPR, je zásadní pro zajištění kompatibility, bezpečnosti a ochrany osobních údajů uživatelů. Bezpečnostní opatření zahrnují robustní ochranu proti

kybernetickým útokům a podvodům, jako je šifrování dat a dvoufaktorová autentizace, což chrání jak data zákazníků, tak integritu celého systému.

2.4 Architektura mikroslužeb (Microservice architecture)

Aplikace eshopu s mikroservisní architekturou je složená z několika samostatných částí, které spolu komunikují pomocí REST API. Každá část aplikace je zodpovědná za určitý aspekt funkcionality a může být vyvíjena, testována a nasazována nezávisle na ostatních částech.



Obrázek 3: *REST architektura*

Klientská část aplikace je zodpovědná za prezentaci a interakci s uživatelem. Frontend komunikuje s backendem a dalšími mikroslužbami pomocí asynchronních HTTP dotazů. Backendová část aplikace poskytuje sadu endpointů (koncových bodů), které představují různé akce nebo operace, které lze provádět s daty. Tyto endpointy jsou definovány pomocí URL adres, například /users pro získání seznamu uživatelů nebo /products/123 pro získání informací o konkrétním produktu. Pro interakci s API jsou využívány standardní HTTP metody jako GET, POST, PUT, DELETE. GET slouží k získání dat, POST k vytvoření nových dat, PUT k aktualizaci existujících dat a DELETE k jejich odstranění. Frontendová část aplikace vytváří HTTP žádosti (requests) na definované endpointy backendu pomocí vhodných HTTP metod. Tyto žádosti mohou obsahovat různé parametry a data, například filtry, stránkování nebo tělo požadavku s informacemi o nových datech. Backend zpracovává

tyto žádosti a vrací odpovědi (responses) ve formátu JSON s odpovídajícími daty. [2], [3], [11]

3 POUŽITÉ NÁSTROJE

Samotný výběr vhodných nástrojů pro vývoj softwaru je klíčovým krokem v procesu tvorby aplikací. V této kapitole se zaměříme na přehled nástrojů, které byly použity při vývoji programu.

3.1 Client side

- **Visual Studio Code:** Visual Studio Code je lehký a výkonný textový editor, který má široké škále funkcí a rozšíření. Poskytuje syntaxové zvýraznění, automatické doplňování kódu, integraci s verzovacími systémy a další užitečné nástroje, které usnadňují vývoj a ladění frontendových aplikací.

- **Google Chrome DevTools**: Google Chrome DevTools jsou součástí prohlížeče Google Chrome a poskytují sadu nástrojů pro ladění webových aplikací. Obsahuje nástroje pro analýzu výkonu, ladění JavaScriptu, editaci CSS a mnoho dalších funkcí.

- **React Developer Tools**: React Developer Tools jsou rozšíření prohlížeče, které umožňuje snadno ladit aplikace vytvořené pomocí knihovny React. Poskytuje možnosti pro prozkoumání a úpravu stavu komponent, analýzu výkonu a kontrolu chyb, což usnadňuje vývoj aplikací.

3.2 Server side

- **IntelliJ IDEA**: IntelliJ IDEA je jedním z předních integrovaných vývojových prostředí (IDE), které je oblíbené mezi vývojáři Java aplikací. Toto vývojové prostředí poskytuje širokou škálu funkcí, jako je refaktorování kódu, zvýrazňování syntaxe, automatické doplňování a integrace s verzovacími systémy.

- **Maven**: Maven byl zvolen jako nástroj pro správu závislostí a sestavení projektu díky své rozšířené komunitě, důkladné dokumentaci a jednoduchosti použití. Jeho deklarativní přístup k definici projektu a závislostí umožňuje snadno a efektivně spravovat a konfigurovat projekty.

- **Git**: Git byl preferovaným verzovacím systémem pro správu zdrojového kódu kvůli jeho flexibilitě a širokému rozšíření. Jeho schopnost podporovat distribuované

pracovní toky, větvení a slučování změn umožňuje efektivní sledování historie změn v projektu.

3.3 Databáze

- **Oracle SQL Developer**: Oracle SQL Developer je integrované vývojové prostředí navržené pro práci s databázemi Oracle. Poskytuje širokou škálu funkcí pro vývoj, správu a ladění databázových dotazů a procedur. Oracle SQL Developer zahrnuje nástroje pro editaci a spouštění SQL dotazů, správu uživatelů a rolí, monitorování výkonu databáze a mnoho dalších funkcí.

- **Oracle Data Modeler**: Oracle Data Modeler je nástroj pro vizuální modelování a návrh databází Oracle. Umožňuje vytvářet a spravovat datové modely pomocí intuitivního grafického rozhraní. Oracle Data Modeler podporuje standardní modelovací techniky, jako je Entity-Relationship (ER) modelování, a poskytuje nástroje pro synchronizaci datových modelů s databázovými schématy, generování skriptů pro vytvoření databáze a mnoho dalších funkcí.

3.1 Testování

- **Postman:** Postman je multifunkční nástroj určený pro testování a ladění API. Jeho hlavním cílem je usnadnit práci s API prostřednictvím intuitivního uživatelského rozhraní. Postman umožňuje vytvářet a odesílat HTTP požadavky na různá API prostřednictvím GUI, což zjednodušuje proces testování a ladění API.

4 POUŽITÉ TECHNOLOGIE

4.1 JavaScript

JavaScript je vysokoúrovňový interpretovaný programovací jazyk, , který je běžně používán pro vývoj webových aplikací. Je to jeden z klíčových jazyků pro tvorbu interaktivního a dynamického obsahu na webových stránkách. JavaScript umožňuje vývojářům přidávat různé funkce a efekty do webových stránek, jako je validace formulářů, animace, manipulace s obsahem stránky a komunikace s webovými službami. JavaScript je dynamicky typovaný jazyk, který podporuje objektově orientované programování, funkcionální programování a asynchronní programování pomocí callback funkcí, promise a asynchronních funkcí. To umožňuje psát čistý a efektivní kód a vytvářet výkonné webové aplikace. [4]

4.2 React

React je populární knihovna JavaScriptu, která je používána k vytváření uživatelských rozhraní. Je vyvíjena společností Facebook a je distribuována jako open-source projekt. Hlavním cílem Reactu je umožnit vývojářům vytvářet interaktivní a dynamické webové aplikace s vysokým výkonem. Jednou z hlavních vlastností Reactu je jeho komponentní architektura. Komponenta v Reactu je znovupoužitelný stavební blok uživatelského rozhraní, který obsahuje vizuální prezentaci a logiku manipulace s daty. Při tvorbě aplikací s Reactem vytváříme množství nezávislých a izolovaných komponent a pak je skládáme do složitých uživatelských rozhraní. Každá aplikace v Reactu má alespoň jednu komponentu, kterou nazýváme kořenovou (root). Tato komponenta reprezentuje vnitřní aplikaci a obsahuje další dceřiné (child) komponenty, takže každá aplikace v Reactu je v podstatě stromem komponent. [555]

Komponenta je obvykle implementována jako JavaScriptová třída, která má nějaký stav a metodu „render()“. Stav zde představuje data, která chceme zobrazit při vykreslení komponenty, a metoda „render()“ je zodpovědná za popis toho, jak by mělo uživatelské rozhraní vypadat. Výstupem této metody je Reactový prvek, což je jednoduchý obyčejný JavaScriptový objekt, který se mapuje na prvek DOM. [5]

DOM (Document Object Model) je reprezentací dokumentu jako stromu prvků. Každý prvek v DOMu je objektem, který má různé vlastnosti a metody a reprezentuje část dokumentu, jako je například element „ “, „<p>“ nebo „“. DOM umožňuje dynamicky manipulovat s obsahem a strukturou dokumentu pomocí JavaScriptu nebo jiných skriptovacích jazyků. Když se vytváří Reactová aplikace, Reactové prvky jsou mapovány na DOM prvky, což umožňuje dynamické vykreslování uživatelského rozhraní na základě změn stavu aplikace. [4], [5]

4.3 Java

V projektu byla využit jazyk **Java**. Java je vysokoúrovňový objektově orientovaný programovací jazyk. Jedná se o platformově nezávislý jazyk, což znamená, že kód napsaný v Javě může být spuštěn na různých operačních systémech, které podporují Java Virtual Machine (JVM). V mém projektu byla Java verze 17 použita pro vývoj backendové části aplikace, která zpracovává a uchovává data, zajišťuje logiku aplikace a komunikaci s databází. Díky své robustnosti, široké podpoře a vysoké škálovatelnosti je Java vhodnou volbou pro náročné a rozsáhlé projekty. [6], [12]

4.4 Lombok

Lombok je knihovna pro jazyk Java, která usnadňuje vývojářům psaní kódu tím, že automaticky generuje opakující se kód za běhu kompilace. Tento nástroj je často využíván v projektech Java, aby snížil množství ručně psaného kódu a zlepšil čitelnost a údržbu aplikací. Hlavní funkcionalitou Lomboku je generování metod jako jsou gettery, settery, equals, hashCode a toString, stejně jako konstruktory, za běhu kompilace. To umožňuje programátorům psát méně kódu, což vede k rychlejšímu a čistšímu vývoji. [8]

K často používaným anotacím v projektu patří:

- **@Data**: Tato anotace automaticky generuje metody toString, equals, hashCode a také gettery a settery pro všechna pole třídy.
- **@AllArgsConstructor**: Tato anotace automaticky generuje konstruktor, který přijímá všechna pole třídy jako argumenty a inicializuje je.

@NoArgsConstructor: Tato anotace generuje konstruktor bez parametrů, který inicializuje všechna pole třídy na výchozí hodnoty. To je užitečné například při vytváření instance třídy bez inicializace jejích polí, například při načítání objektů z databáze nebo deserializaci JSON dat. [7], [8]

4.5 Spring Boot

Spring Boot je nadstavba nad Spring Frameworkem, která značně zjednodušuje proces vytváření aplikací v jazyce Java. Narozdíl od Spring Frameworku, který vyžaduje ruční konfiguraci pomocí XML nebo anotací a explicitní definování závislostí, Spring Boot poskytuje stanovené konvence a automatickou konfiguraci, což umožňuje vytvářet aplikace s minimálním množstvím konfiguračního kódu a s automatickým přidáváním závislostí na základě potřeb aplikace. Dále obsahuje integrovaný vývojový server, který umožňuje spouštět aplikace přímo z vývojového prostředí. Spring Boot také usnadňuje aktualizaci projektů na novější verze Spring Frameworku a jeho závislostí. To přispívá k udržení aplikací v synchronizaci s nejnovějšími verzemi Spring Frameworku a jeho závislostí, čímž zajišťuje, že aplikace zůstávají aktualizované a bezpečné. [9], [10]

4.6 Spring framework

Spring Framework je jedním z nejpopulárnějších a rozsáhlých frameworků pro vývoj aplikací v jazyce Java. Jeho flexibilita, škálovatelnost a rozsáhlý ekosystém knihoven

poskytují prostředky pro vytváření široké škály aplikací, od jednoduchých webových aplikací až po rozsáhlé podnikové systémy.

Hlavním cílem Spring Frameworku je zjednodušení vývoje aplikací a snížení množství boilerplate kódu (opakovaně používaný kód nebo vzorová šablona). Jedním z klíčových principů Springu je tzv. "Dependency Injection" (DI), neboli injekce závislostí, která umožňuje snadnou správu závislostí mezi komponentami aplikace. DI zjednodušuje testování a zvyšuje modularitu aplikace tím, že odděluje vytváření objektů od jejich použití. Dalším základním prvkem Springu je "Aspect-Oriented Programming" (AOP), což je technika programování, která umožňuje oddělit různé aspekty funkcionality aplikace, jako jsou například logování, transakce nebo zabezpečení, od hlavního kódu aplikace. To zvyšuje modularitu a opět usnadňuje testování a údržbu aplikace. [1], [11]

Spring Framework se skládá z několika modulů, z nichž každý poskytuje různé funkce a nástroje pro vývoj aplikací. Mezi nejznámější moduly patří:

- **Spring Core Container**: Obsahuje základní funkce frameworku, jako je správa životního cyklu objektů, konfigurace pomocí XML, Java nebo anotací a podpora pro Dependency Injection.
- **Spring MVC**: Poskytuje framework pro vývoj webových aplikací založených na architektuře MVC (Model-View-Controller). Umožňuje snadnou tvorbu webových stránek a RESTful služeb.
- **Spring Data**: Nabízí abstrakci pro práci s daty a databázemi. Podporuje různé typy databází a poskytuje nástroje pro snadnou integraci s persistentní vrstvou aplikace.
- **Spring Security**: Zajišťuje zabezpečení aplikace pomocí autentizace, autorizace a ochrany proti různým útokům.
- **Spring Boot**: Je nadstavbou nad Spring Frameworkem, která usnadňuje a urychluje proces vytváření aplikací tím, že automaticky konfiguruje mnoho běžných nastavení a knihoven. [11], [12]

V Spring Frameworku anotace jsou metadata, která poskytují dodatečné informace o třídách, metodách nebo jiných prvcích aplikace. Tyto anotace se používají k deklarativní konfiguraci aplikace a k definici určitých aspektů chování aplikace.

- **@Autowired**: Anotace @Autowired je součástí Spring Frameworku a slouží k automatickému vkládání závislostí do tříd, tedy injekci závislostí. Když třída obsahuje atribut s anotací @Autowired, Spring se automaticky pokusí najít vhodnou instanci tohoto typu a vložit ji do daného atributu.

- **@RestController**: Tato anotace označuje třídu jako RESTful controller v rámci Spring Frameworku. To znamená, že třída je schopna přijímat HTTP požadavky a vracet odpovědi ve formátu JSON nebo XML. Jedná se o kombinaci anotace @Controller a @ResponseBody.

- **@Controller**: Tato anotace označuje třídu jako součást MVC architektury ve Spring Frameworku. Třídy označené touto anotací obvykle obsahují metody, které slouží k obsluze HTTP požadavků a vracení odpovědí. Metody v těchto třídách obvykle zpracovávají data a vykreslují odpovídající uživatelské rozhraní nebo předávají data na další zpracování.

- **@ResponseBody**: Tato anotace se obvykle používá v kombinaci s anotací @Controller. Označuje, že návratová hodnota metody má být přímo zapsána do těla HTTP odpovědi, aniž by byla renderována do HTML šablony. To znamená, že data vrácená z metody budou převedena na formát, který je vhodný pro přenos (například JSON nebo XML) a přímo odeslána klientovi.

- **@Service**: Tato anotace se používá pro označení služebních (service) komponent v architektuře Spring. Služební třídy obvykle obsahují business logiku aplikace a jsou zodpovědné za zpracování dat.

- **@Repository**: Tato anotace se používá pro označení tříd, které se starají o práci s databází nebo persistentní vrstvou aplikace. Třídy označené touto anotací obvykle obsahují metody pro provádění CRUD operací nad daty.

- **@GetMapping, @PostMapping, @PutMapping, @DeleteMapping**: Tyto anotace specifikují, které metody kontroleru mají být volány při určitých HTTP požadavcích. Například @GetMapping označuje metodu, která má být volána pouze v případě, že je přijat GET požadavek, zatímco @PostMapping označuje metodu pro zpracování POST požadavků. Analogicky se chovají i @PutMapping a @DeleteMapping pro PUT a DELETE požadavky. [11], [12]

4.7 SQL

SQL (Structured Query Language) je standardizovaný jazyk pro práci s relačními databázemi. Byl vyvinut jako prostředek pro vytváření, manipulaci a dotazování na data uložená v relačních databázových systémech.

4.7.1 Tabulky

Jednou z nejzákladnějších konstrukcí v SQL jsou tabulky, které slouží k ukládání a organizaci dat. Tabulky jsou základními prvky relačních databází a skládají se z řádků a sloupců. Každá tabulka má svůj název a definici struktury, která určuje, jaká data mohou být v tabulce uložena. Struktura tabulky je definována souborem sloupců, přičemž každý sloupec má svůj název a datový typ. Datový typ určuje typ dat, které může být v daném sloupci uloženo, například celé číslo (INTEGER), textový řetězec (VARCHAR), datum (DATE) atd. [13]

4.7.2 Vztahy

V relačních databázích jsou vztahy klíčovým prvkem pro propojení dat uložených v různých tabulkách. Vztahy určují, jak jsou záznamy v jedné tabulce spojeny s záznamy v jiné tabulce. Existují různé typy vztahů, které mohou existovat mezi tabulkami:

- **1:1** (One-to-One): Každý záznam v jedné tabulce je spojen s právě jedním záznamem v druhé tabulce a naopak. Tento typ vztahu se často používá, když je mezi dvěma entitami přímý a jednoznačný vztah.
- **1:N** (One-to-Many): V tomto typu vztahu každý záznam v jedné tabulce může být spojen s jedním nebo více záznamy v druhé tabulce, ale každý záznam v druhé tabulce je spojen právě s jedním záznamem v první tabulce.
- **M:N** (Many-to-Many): V tomto typu vztahu každý záznam v jedné tabulce může být spojen s mnoha záznamy v druhé tabulce a naopak. Tento typ vztahu vyžaduje vytvoření spojovací tabulky, která obsahuje cizí klíče z obou tabulek a umožňuje propojení mezi nimi. [14], [15]

4.7.3 Příkazy

SQL poskytuje sadu příkazů pro definování databázových struktur, manipulaci s daty a získávání informací z databází. Základní operace v SQL zahrnují:

- **DDL** (Data Definition Language): Tento typ příkazů se používá pro definování struktury databáze, jako jsou tabulky, indexy, pohledy, uložené procedury atd.

Příklady zahrnují CREATE pro vytvoření nových prvků a ALTER pro úpravu existujících struktur.

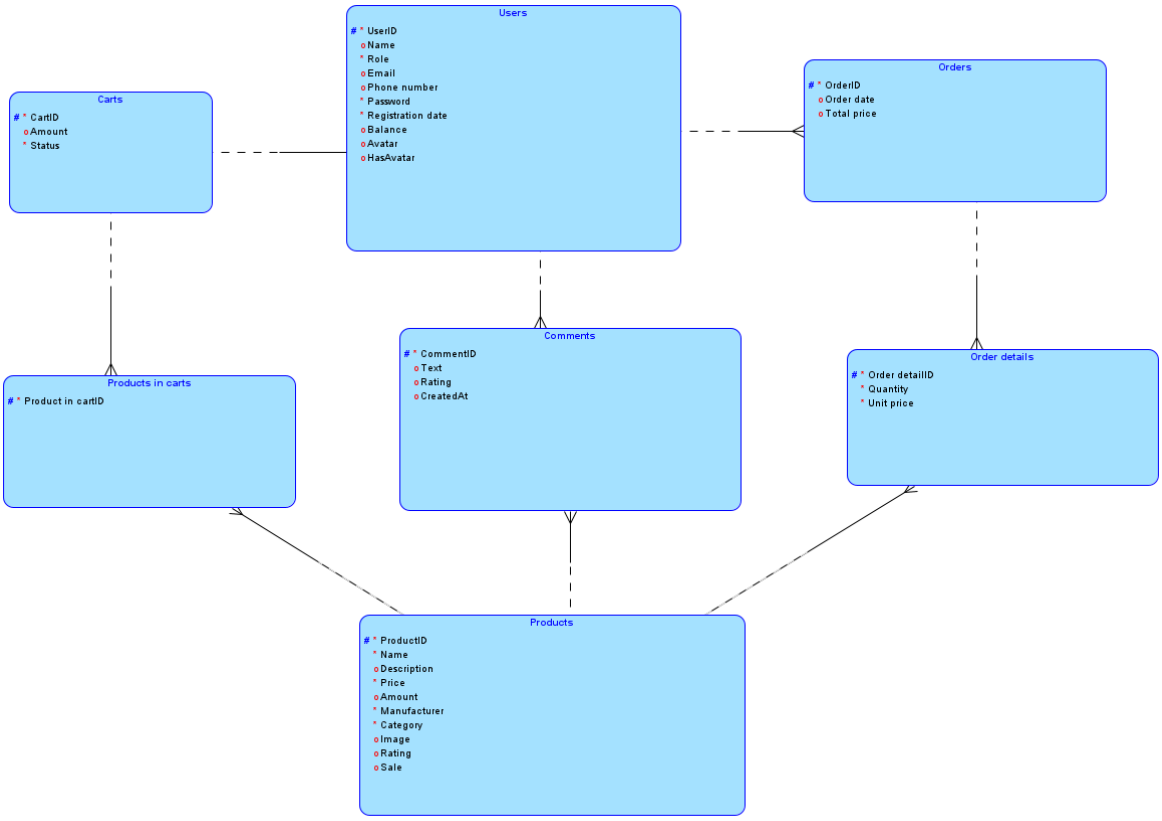
- **DML** (Data Manipulation Language): DML příkazy umožňují manipulaci s daty v databázi. Patří sem příkazy INSERT pro vkládání nových záznamů, UPDATE pro aktualizaci existujících záznamů a DELETE pro mazání záznamů.
- **DQL** (Data Query Language): DQL příkazy se používají k dotazování se na data v databázi pomocí příkazu SELECT. Tyto dotazy umožňují vybrat konkrétní sloupce, filtrovat výsledky podle určitých podmínek a provádět agregační funkce.
- **DCL** (Data Control Language): DCL příkazy se používají pro řízení přístupu k datům v databázi. Patří sem příkazy GRANT pro udělování přístupových práv a REVOKE pro zrušení přístupových práv. [13], [15]

5 DATABÁZE

Databáze představuje klíčový prvek v mnoha informačních systémech, včetně elektronických obchodů. Aplikace využívá relační databázi Oracle pro ukládání a správu dat. V této kapitole se zaměříme na zkoumání všech složených prvků.

5.1 Logický model

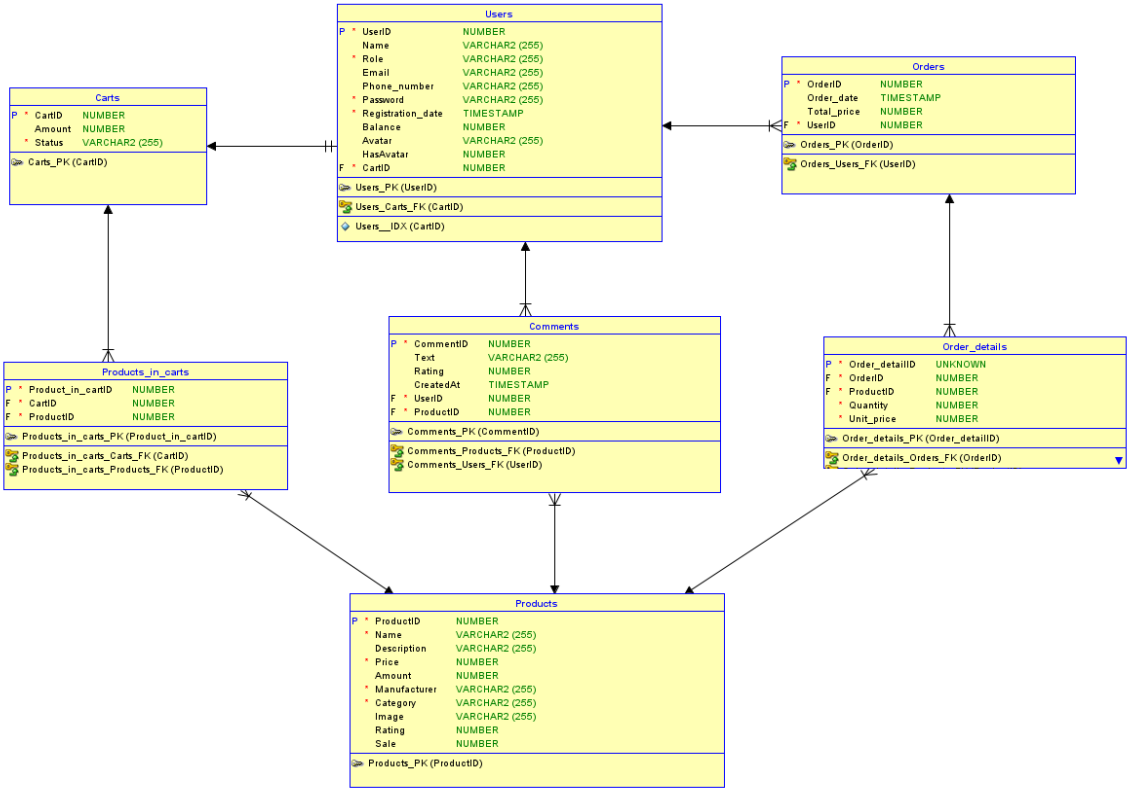
Logický model databáze popisuje strukturu a vztahy mezi daty, které jsou uloženy v databázi, bez ohledu na konkrétní technické detaily implementace. V našem logickém modelu jsou entity a vztahy mezi nimi definovány jako logické koncepty, což umožňuje plánovat a navrhovat strukturu databáze bez závislosti na konkrétním DBMS (Database Management System). [14]



Obrázek 4: *logický model*

5.2 ER diagram

ER (Entity-Relationship) diagram je grafický nástroj pro vizualizaci struktury a vztahů mezi entitami v databázi. ER diagram obsahuje entity, atributy a vztahy mezi nimi. Entita představuje objekt, o kterém ukládáme informace, atributy jsou vlastnosti entit a vztahy určují, jak jsou entity propojeny. ER diagram nám poskytuje přehledný způsob vizualizace struktury databáze a je užitečný pro plánování, návrh a porozumění datové architektuře aplikace. [14]



Obrázek 5: *ER diagram*

5.3 Tabulky

Tabulky představují základní stavební bloky relační databáze, ve kterých jsou uchovávána data organizovaná do řádků a sloupců. Každá tabulka reprezentuje určitý typ informace a obsahuje atributy, které definují vlastnosti této informace. V této části se podrobně podíváme na jednotlivé tabulky v naší databázi a jejich strukturu, atributy a vztahy mezi nimi. [13]

5.3.1 USERS

Tabulka Users uchovává informace o uživatelích.

- *UserID* - primární klíč identifikující uživatele.
- *Username* - textový sloupec obsahující jméno uživatele.
- *Role* - textový sloupec obsahující roli uživatele.
- *Email* - textový sloupec obsahující e-mailovou adresu uživatele.
- *Phone_number* - textový sloupec obsahující telefonní číslo uživatele.
- *Password* - textový sloupec obsahující heslo uživatele.

- *Registration_date* - sloupec obsahující datum registrace uživatele.
- *Balance* - číselný sloupec obsahující zůstatek na účtu uživatele.
- *Avatar* - textový sloupec obsahující URL avatara uživatele.
- *Hasavatar* - číselný sloupec označující, zda uživatel má avatar.
- *CartID* - cizí klíč, který odkazuje na nákupní koš uživatele.

Relace:

- *Carts*: 1:1, každý uživatel má pouze 1 košík.
- *Orders*: 1:M, každý uživatel může mít několik objednávek.
- *Comments*: 1:M, každý uživatel může mít několik komentářů.

5.3.2 PRODUCTS

Tabulka Products uchovává informace o produktech.

- *ProductID* - primární klíč identifikující produkt.
- *Name* - textový sloupec obsahující název produktu.
- *Description* - textový sloupec obsahující popis produktu.
- *Price* - číselný sloupec obsahující cenu produktu.
- *Amount* - číselný sloupec obsahující dostupné množství produktu.
- *Manufacturer* - textový sloupec obsahující výrobce produktu.
- *Category* - textový sloupec obsahující kategorii produktu.
- *Image* - textový sloupec obsahující URL obrázku produktu.
- *Rating* - číselný sloupec obsahující hodnocení produktu.
- *Sale* - číselný sloupec obsahující slevu produktu.

Relace:

- *Comments*: 1:M, každý product může mít několik komentářů.
- *Products_in_carts*: 1:M, každý produkt může být v několika koších.

- *Order_details*: 1:M, každý produkt může být v několika objednávkách.

5.3.3 COMMENTS

Tabulka Comments uchovává komentáře uživatelů k produktům.

- *CommentID* - primární klíč identifikující komentář.
- *Text* - textový sloupec obsahující samotný text komentáře.
- *Rating* - číselný sloupec obsahující hodnocení komentáře.
- *CreatedAt* - sloupec obsahující datum a čas vytvoření komentáře.
- *UserID* - cizí klíč, který odkazuje na uživatele, který vytvořil komentář.
- *ProductID* - cizí klíč, který odkazuje na produkt, ke kterému je komentář připojen.

Relace:

- *Products*: 1:M, každý product může mít několik komentářů.
- *Users*: 1:M, každý uživatel může mít několik komentářů.

5.3.4 CARTS

Tabulka Carts uchovává informace o nákupních koších.

- *CartID* - primární klíč identifikující nákupní koš.
- *Amount* - číselný sloupec obsahující celkovou částku koše.
- *Status* - textový sloupec obsahující stav nákupního koše.

Relace:

- *Users:* 1:1, každý uživatel má pouze 1 košík.
- *Product_in_carts:* M:1, každý koš může obsahovat několik produktů.

5.3.5 PRODUCTS_IN_CARTS

Tabulka Products_in_carts uchovává vazby mezi produkty a nákupními košíky.

- *Product_in_cartID* - primární klíč identifikující product v koši.
- *CartID* - cizí klíč, který odkazuje na nákupní koš.
- *ProductID* - cizí klíč, který odkazuje na produkt.

Relace:

- *Carts*: M:1, jeden produkt může být obsažen v několika koších.
- *Products*: M:1, jeden koš může obsahovat několik produktů.

5.3.6 ORDERS

Tabulka Orders uchovává informace o objednávkách.

- *Orderid* - primární klíč identifikující objednávku.
- *Order_date* - sloupec obsahující datum a čas vytvoření objednávky.
- *Total_price* - číselný sloupec obsahující celkovou cenu objednávky.
- *UserID* - cizí klíč, který odkazuje na uživatele, který vytvořil objednávku.

Relace:

- *Users*: 1:M, každý uživatel může mít několik objednávek.
- *Order_details*: 1:M, každá objednávka může obsahovat několik položek.

5.3.7 ORDER_DETAILS

Tabulka order_details uchovává detaily o položkách v objednávkách.

- *OrderID* - cizí klíč, který odkazuje na objednávku, ke které patří tato položka.
- *ProductID* - cizí klíč, který odkazuje na produkt této položky.
- *Quantity* - číselný sloupec obsahující množství této položky v objednávce.
- *Unit_price* - číselný sloupec obsahující jednotkovou cenu této položky.

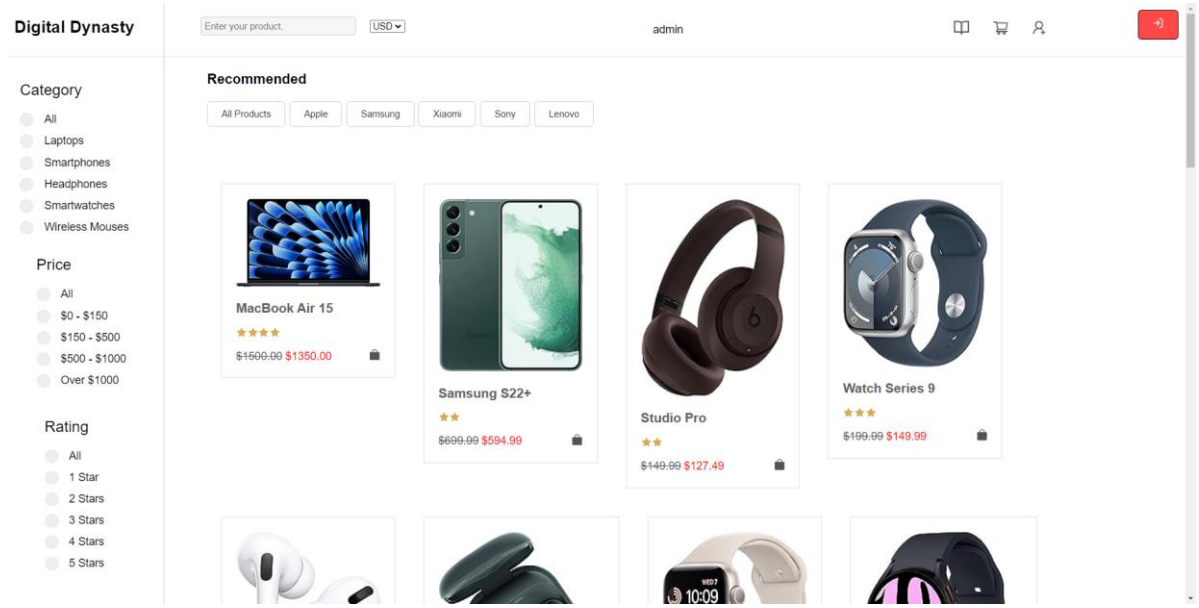
Relace:

- *Orders*: 1:M, každá objednávka může obsahovat několik položek.
- *Products*: M:1, každá položka objednávky odkazuje na jeden produkt.

6 DEMONSTRACE APLIKACE

6.1 Domovská stránka

Aplikace začíná na domovské stránce, která je prvním místem, kam nový uživatel přistane. Na domovské stránce se uživatel seznámí s nabídkou produktů prostřednictvím atraktivně zobrazených karet. Tyto karty obsahují klíčové informace, jako jsou obrázek produktu, název, cena a hodnocení, což umožňuje uživatelům snadno vybrat požadovaný produkt.



Obrázek 6: *domovská stranka*

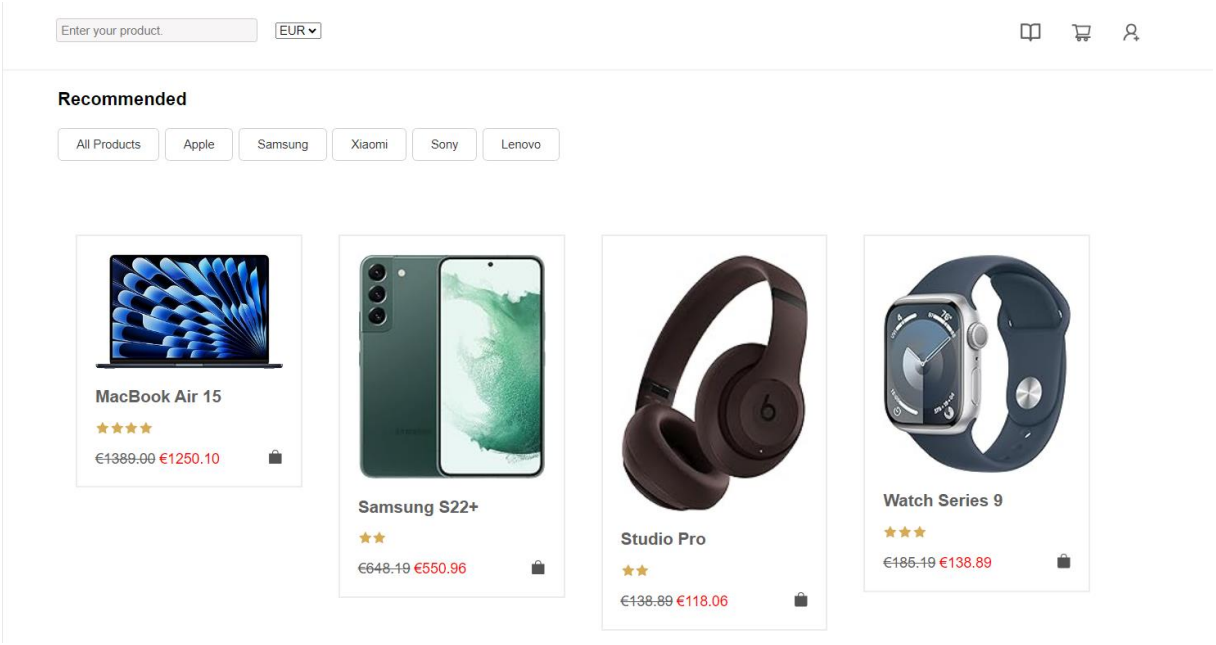
Díky filtrům umístěným na levé a horní části stránky mají uživatelé možnost upřesnit své hledání a najít produkty podle svých preferencí. Filtry umožňují výběr produktů podle kategorie, ceny, výrobce a hodnocení. Uživatelé mohou použít více filtrů současně. Filtrace se provádí přímo na frontendu, což umožňuje okamžitou reakci a žádné zpoždění při filtrování produktů. Díky této funkci není potřeba odesílat žádné dotazy na server, což šetří čas a zdroje.

V horní části stránky se nachází navigační záhlaví obsahující pole pro vyhledávání produktů podle názvu. Po zadání hledaného výrazu se zobrazí kombinované pole, ve kterém lze vybrat požadovanou měnu (výchozí hodnota je USD, ale lze změnit na EUR nebo CZK). Změna hodnoty v tomto kombinovaném poli provede žádost o aktuální směnný kurz prostřednictvím otevřeného bankovního API a aktualizuje měnu pro nákupy produktů na základě získaných informací. Dále na této části stránky je zobrazeno uživatelské jméno přihlášeného uživatele. Následuje řada tlačítek umožňujících přístup k různým funkcím aplikace, jako jsou objednávky, košík, osobní profil, autentizace a autorizace.

```
useEffect(() => {
  if (currency !== "USD") {
    fetch(`https://api.frankfurter.app/latest?amount=1&from=USD&to=${currency}`)
      .then((response) => response.json())
      .then((data) => {
        setConversionRate(data.rates[currency]);
      })
      .catch((error) => console.error("Error fetching currency conversion:", error));
  } else {
    setConversionRate(1); // Set conversion rate to 1 if currency is USD
  }
}, [currency]);
```

Obrázek 7: *příklad kódu pro získání kurzu měny*

Tento kód odesílá požadavek na adresu „https://api.frankfurter.app/latest?amount=1&from=USD&to=%s“, kde „%s“ představuje novou měnu z comboboxu. V odpovědi přichází skutečný kurz měny a ceny se přizpůsobí tomuto kurzovému poměru.



Obrázek 8: *příklad produktů s novou měnou*

6.2 Autentizace

Autentizace v projektu je klíčovou funkcí pro zajištění plného přístupu k všem funkcím webové stránky. Uživatelé se musí registrovat nebo přihlásit, aby mohli využívat veškeré možnosti stránky. Kliknutím na červené tlačítko v pravém horním rohu uživatelé budou přesměrováni na stránku s autentizací. Zde jsou k dispozici tři tlačítka – “Login”, “Register” a “Continue without login”. Tlačítka “Register” a “Login” otevírají stránky s polem pro zadání

uživatelského jména a hesla. “Continue without login” vrátí uživatele zpět na domovskou stránku.

Obrázek 9: *okno pro registraci*

Pro zaregistrování nového uživatelského účtu frontendová část aplikace zasílá požadavek na určený backendový endpoint s názvem „/registration“, který obsahuje jméno a heslo, jež uživatel zadal. Nejprve se provádí kontrola, zda zadané uživatelské jméno není již obsazené. V případě, že existuje uživatel s tímto jménem, je frontendové aplikaci vrácena odpovídající zpráva. Poté je heslo zašifrováno pomocí algoritmu bcrypt a uživatelská data jsou uložena do databáze.

```java
@Service
@RequiredArgsConstructor
public class UserService {

    private final UserDao userDao;
    private final BCryptPasswordEncoder passwordEncoder;
    1 usage
    public void registration(RegistrationUserRequest user) {
        if(userDao.checkIsUserExistByUsername(user.getUsername())){
            throw new RuntimeException("Choose another username. This username already exists!");
        }
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        userDao.addNewUser(user);
    }

    no usages
    public UserDto getUserById(Integer userId) { return userDao.getUserById(userId).toUserDto(); }
```

Obrázek 10: *příklad kódu pro registraci*

Pro přístup do svého účtu uživatel zadává do frontendové aplikace své přihlašovací údaje, které jsou pak odeslány na backendový server na specifickou adresu „/login“. Pro ověření identity a oprávnění uživatelů v aplikaci eshopu je použit framework Spring Security. Po úspěšném zpracování přihlašovacího požadavku je uživateli zaslána zpráva o úspěchu a spolu s ní i JSON Web Token (JWT), který slouží k ověření a autorizaci jeho dalších akcí.

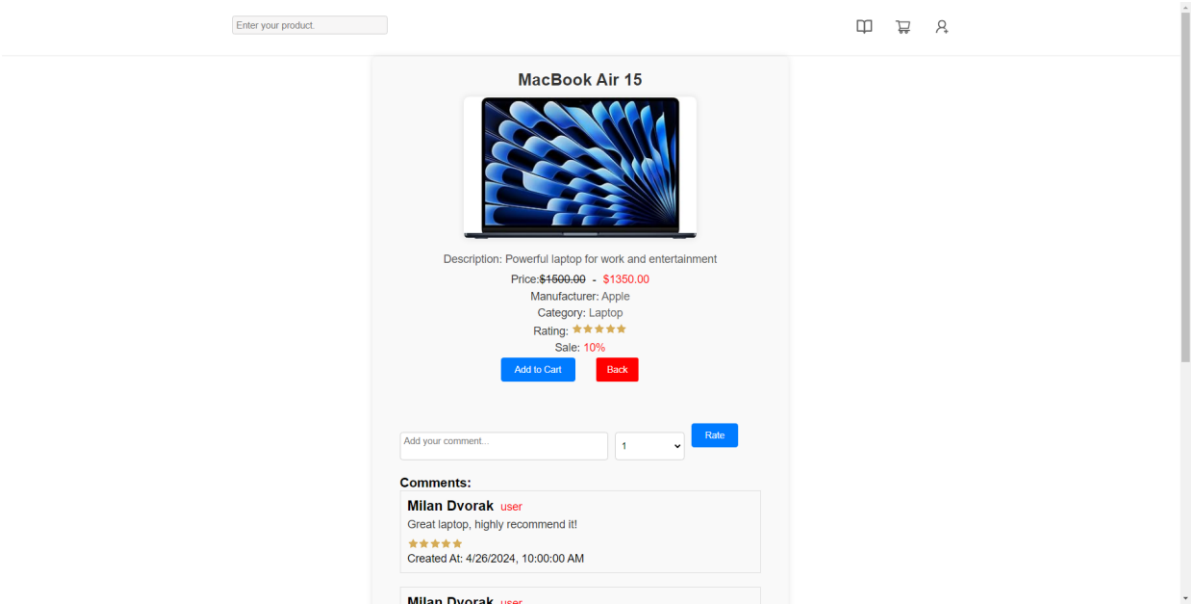
```java
@RestController
@RequestMapping(value = "/api/auth")
@RequiredArgsConstructor
public class AuthController {

    private final AuthenticationManager authenticationManager;
    private final JwtTokenProvider jwtTokenProvider;
    private final UserService userService;


    @PostMapping(value = "/login")
    public ResponseEntity<UserDto> login(@RequestBody AuthenticationRequest request, HttpSession session) {
        try {
            Authentication authenticate = authenticationManager
                    .authenticate(new UsernamePasswordAuthenticationToken(request.getUsername(), request.getPassword()));

            User user = (User) authenticate.getPrincipal();

            // Store user info in the session
            session.setAttribute( s: "userId", user.getId());
            session.setAttribute( s: "username", user.getUsername());
            System.out.println("Gocha");

            return ResponseEntity.ok()
                    .header(HttpHeaders.AUTHORIZATION, jwtTokenProvider.generateAccessToken(user))
                    .body(user.toUserDto());
        } catch (BadCredentialsException ex) {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();
        }
    }
```

Obrázek 11: *příklad kódu pro autentizace*

Po úspěšném přihlášení uživatele frontendová aplikace získává JWT token, který je poté uchováván a přikládán ke každému uživatelskému požadavku pro autorizaci. Pro kontrolu a ověření oprávnění je odpovědný JwtTokenFilter, který zkontroluje platnost a správnost JWT tokenu přiloženého uživatelem.

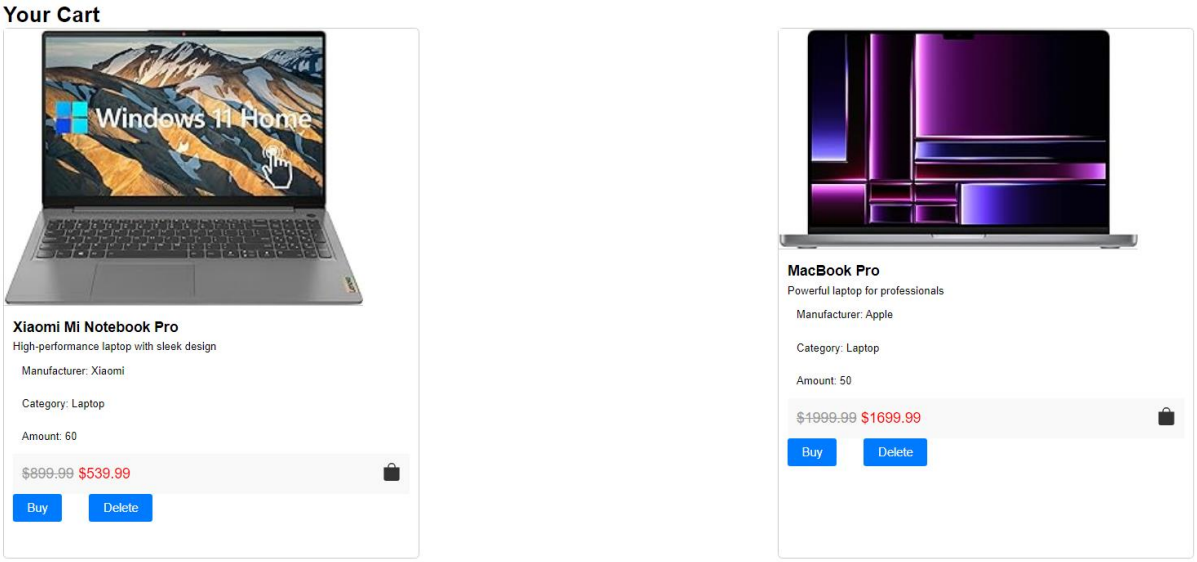
6.3 Nákupy

Po kliknutí na kartu produktu na domovské stránce se otevře stránka produktu s veškerými detaily. Kliknutím na tlačítko "Add to cart" uživatel může přidat zboží do košíku, odkud bude později provedena platba.



Obrázek 12: *stránka produktu*

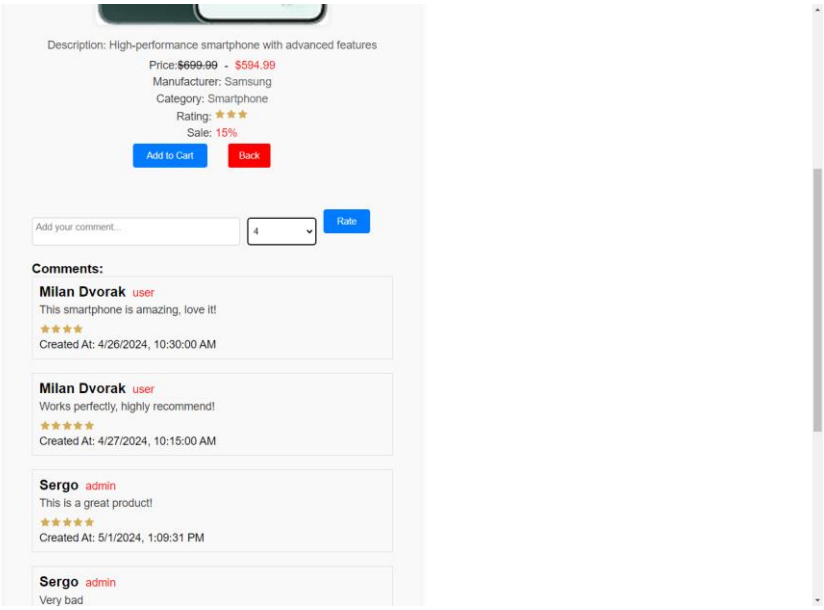
Pro dokončení nákupu je nutné přejít na stránku "Cart", kde jsou zobrazeny všechny produkty přidané uživatelem. Odstranění zboží z košíku je možné pomocí tlačítka "Delete". Po kliknutí na tlačítko "Buy" a v případě dostatečného množství prostředků na virtuálním účtu bude nákup proveden. Zboží bude odebráno z košíku, množství zboží na virtuálním skladě se sníží a prostředky uživatele budou odečteny. Informace o objednávce bude také uložena do databáze v tabulce "Orders".



Obrázek 13: stránka košíku

6.4 Komentáře a hodnocení

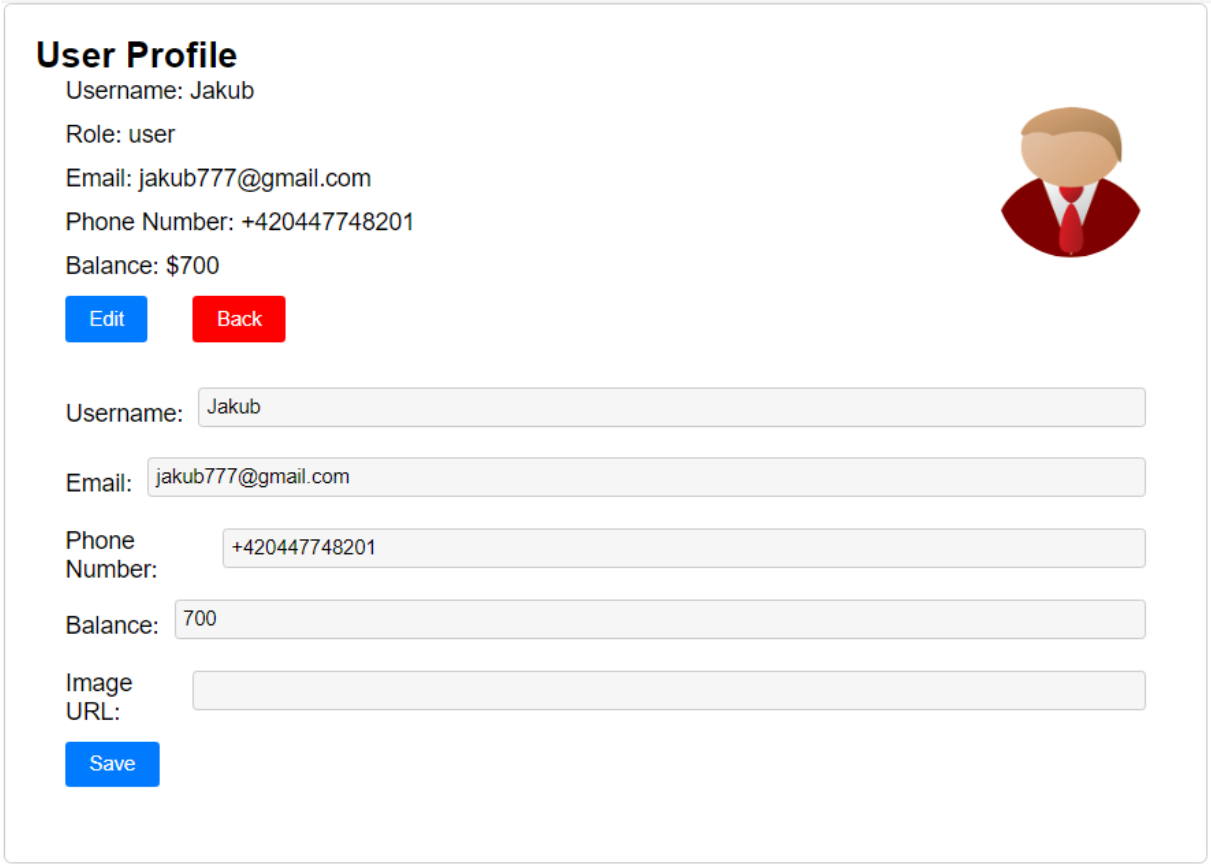
Identifikovaný uživatel má možnost přidávat komentáře a hodnocení k produktům. Na stránce s detaily produktu je k dispozici pole pro zadání textu komentáře a combobox pro hodnocení produktu. Po stisknutí tlačítka "Rate" je odeslán požadavek na server, kde je nový komentář přidán do databáze. Po přidání nového komentáře je aktualizováno hodnocení produktu. Nové hodnocení produktu se rovná průměrné hodnotě všech hodnocení přidělených k tomuto produktu.



Obrázek 14: komentáře na stránce produktu

6.5 Uživatelský panel

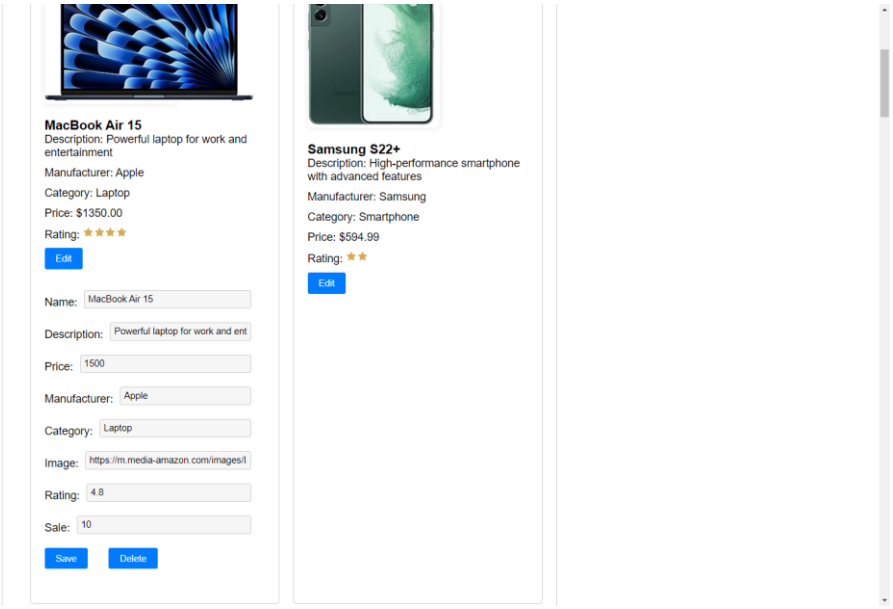
Uživatel s přístupovým úrovní "user" má přístup k osobnímu účtu. V tomto panelu uživatel vidí své osobní údaje a může je upravovat pomocí tlačítka "Edit". Mezi tyto údaje patří uživatelské jméno, email, telefonní číslo a URL obrázku. Pokud není uvedena adresa URL obrázku, bude zobrazen výchozí obrázek uživatele.



Obrázek 15: *osobní profil uživatele*

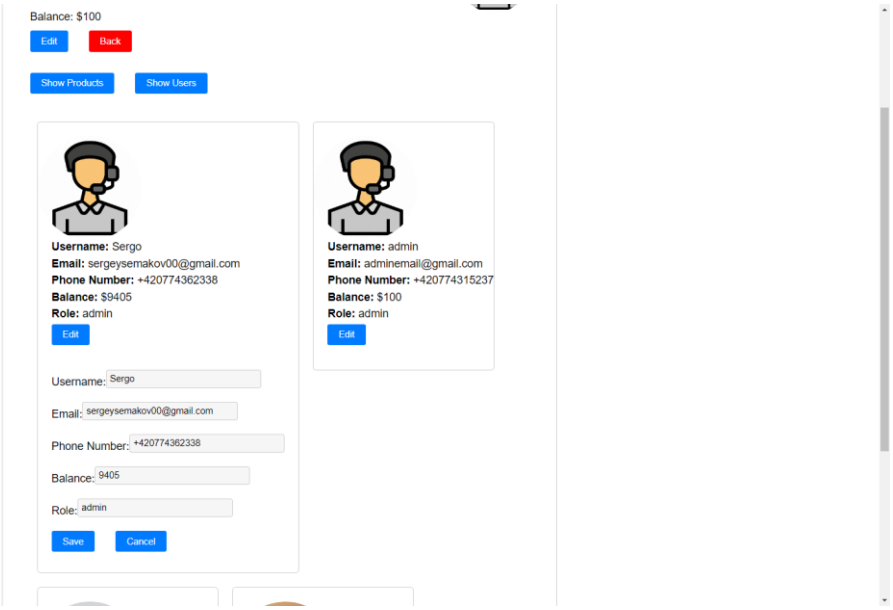
6.6 Administrátorský panel

Administrátor má všechna stejná oprávnění jako běžný uživatel, ale navíc má možnost spravovat data produktů a uživatelů. Kliknutím na tlačítko "Show products" se zobrazí všechny produkty s možností úpravy. Po kliknutí na tlačítko "Upravit" vedle produktu se zobrazí pole pro zadání nových hodnot parametrů tohoto produktu. Pro editaci produktu jsou k dispozici následující pole: název, popis, cena, množství, výrobce, kategorie, obrázek, hodnocení, sleva. Kliknutím na tlačítko "Uložit" bude odeslán požadavek na server, který provede změnu produktu v databázi. Tlačítko "Delete" umožňuje úplné odstranění položky z obchodu.



Obrázek 16: *administrátorský panel pro správu dat produktů*

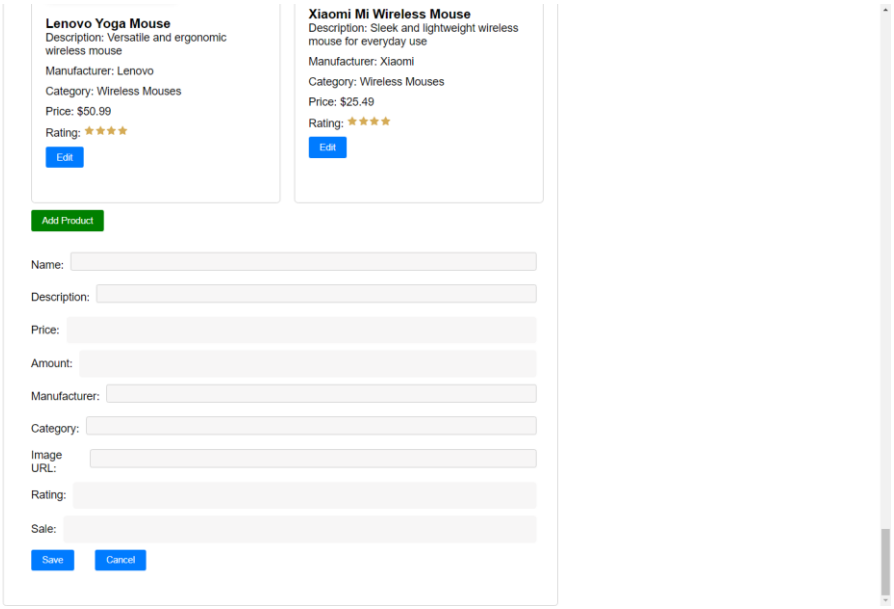
Kliknutím na tlačítko "Show users" se administrátorovi zobrazí všichni existující uživatelé. Stejně jako u produktů, tlačítko "Edit" vedle uživatele umožňuje editovat jeho osobní informace. Pole pro úpravu obsahují následující údaje: uživatelské jméno, email, telefonní číslo, zůstatek a role. Kliknutím na tlačítko "Save" bude odeslán požadavek na server, který provede změny uživatele v databázi. Tlačítko "Delete" umožňuje úplné odstranění uživatele.



Obrázek 17: *administrátorský panel pro správu dat uživatelů*

Po stisknutí tlačítka "Add product" se otevřou prázdná pole pro vyplnění parametrů nového produktu. Mezi ně patří název, popis, cena, množství, výrobce, kategorie, URL obrázku, hodnocení a sleva. Stisknutím tlačítka "Cancel" se vytváření nového produktu zruší.

Stisknutím tlačítka "Save" se odešle POST požadavek na backend a nový produkt se uloží do databáze.



Obrázek 18: *administrátorský panel pro přidání nového produktu*

6.7 Modifikace projektu

Během vývoje internetového obchodu jsem se zaměřil na implementaci všech klíčových funkcionalit nezbytných pro jeho správné fungování. Je důležité poznamenat, že vzhledem k omezeným možnostem obsahu jsem nezahrnul některé prvky, které by mohly přinést další přínosy aplikaci a zlepšit uživatelský zážitek. Tyto prvky by v budoucnu mohly představovat cenná vylepšení a rozšíření naší platformy.

- Personalizované doporučení produktů: Implementace algoritmů doporučení, které by analyzovaly chování uživatelů a na základě toho jim nabídly personalizované doporučení produktů, by mohla zvýšit míru konverze a uživatelskou spokojenost.
- Responsivní design: Vytvoření responzivního designu umožní naší aplikaci přizpůsobit se různým zařízením a obrazovkám, což je dnes velmi důležitá vlastnost. I když implementace tohoto prvku může být časově náročná, může to přinést značné vylepšení uživatelského zážitku.
- Rozšíření platebních možností: Přidání dalších platebních možností, jako jsou platební brány, mobilní platby nebo platby přes internetové peněženky, by mohlo zvýšit pohodlí a dostupnost pro naše zákazníky.

ZÁVĚR

Vytvoření webové aplikace elektronického obchodu představuje důležitý krok v prostředí stále se rozvíjejícího e-commerce trhu. Tato práce si klade za cíl vytvořit funkční a uživatelsky přívětivou platformu, která usnadní nakupování a správu produktů online.

Během průzkumu tématu elektronických obchodů jsme identifikovali klíčové faktory úspěchu a trendy v oblasti e-commerce. Tyto poznatky nám sloužily jako základ pro návrh a implementaci webové aplikace.

V rámci praktické části práce jsem se zaměřil na konkrétní implementaci navrženého elektronického obchodu. Sledoval jsem pečlivě definované cíle a vytvořil jsem systém, který splňuje potřeby uživatelů a poskytuje jednoduché a efektivní prostředí pro nákup zboží.

Díky použití moderních technologií jako JavaScript, React, Java a Spring Boot jsem dosáhl stabilního a funkčního systému. Integrace s databází Oracle umožňuje efektivní správu dat a snadnou manipulaci s informacemi.

Celkově lze říci, že cíle této práce byly úspěšně dosaženy. Vytvořená webová aplikace elektronického obchodu představuje robustní platformu pro online nakupování. Její rozšiřitelnost a možnost dalšího vývoje nabízejí potenciál pro budoucí vylepšení a rozšíření funkcionalit.

POUŽITÁ LITERATURA

[1] WALLS, Craig. Spring in Action. Sebastopol: O'Reilly Media, 2018. ISBN 978-1-61729-494-5.

[2] RICHARDSON, Leonard. RESTful Web APIs: Services for a Changing World. O'Reilly Media, 2013. ISBN 9781449359720.

[3] ALLAMARAJU, Subbu. RESTful Web Services Cookbook: Solutions for Improving Scalability and Simplicity. O'Reilly Media, 2010. ISBN 0596801688.

[4] HAVERBEKE, Marijn. Eloquent JavaScript, 3rd Edition: A Modern Introduction to Programming. 3. No Starch Press, 2018. ISBN 1593279507.

[5] CHINNATHAMBI, Kirupa. Learning React: A Hands-On Guide to Building Web Applications Using React and Redux. 2. Addison-Wesley Professional, 2018. ISBN 013484355X.

[6] SCHILDT, Herbert. Java: A Beginner's Guide, Eighth Edition. 8. McGraw Hill, 2018. ISBN 1260440214.

[7] KAMATH, Uday a Krishna CHOPPELLA. Mastering Java Machine Learning: A Java developer's guide to implementing machine learning and big data architectures. Packt Publishing, 2017. ISBN 1785880519.

[8] Project Lombok. Lombok features [online]. c2009-2024 [cit. 2024-04-27]. Dostupné z: https://projectlombok.org/features/all

[9] Spring.io. Spring Boot [online]. c2024 [cit. 2024-04-27]. Dostupné z: https://spring.io/projects/spring-boot

[10] WALLS, Craig. Spring Boot in Action. Manning, 2016. ISBN 978-1617292545.

[11] Spring.io. Building REST services with Spring [online]. c2022 [cit. 2022-04-23]. Dostupné z: https://docs.spring.io/spring-boot/docs/current/reference/html/documentation.html

[12] COSMINA, Iuliana, Rob HARROP, Chris SCHAEFER a Clarence HO. Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools. 5. Apress, 2017. ISBN 9781484228074.

[13] TAYLOR, Allen G. SQL for Dummies. 8. For Dummies, 2013. ISBN 1118607961.

[14] GARCIA-MOLINA, Hector, Jeffrey ULLMAN a Jennifer WIDOM. Database Systems: The Complete Book. 2. Pearson, 2008. ISBN 0131873253.

[15] BEAULIEU, Alan. Learning SQL: Generate, Manipulate, and Retrieve Data. 3. O'Reilly Media, 2020. ISBN 1492057614.

SEZNAM PŘÍLOH

Příloha A – DDL script pro vytvoření databáze.

Příloha B – zdrojový kód frontendu. Je veřejně dostupný z GitHubu: https://github.com/Butyratino/Bakalarska-prace-Semakou-2024/

Příloha C – zdrojový kód backendu. Je veřejně dostupný z GitHubu: https://github.com/Butyratino/Bakalarska-prace-Semakou-2024/