

UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY  
A INFORMATIKY

DIPLOMOVÁ PRÁCE

2025

Bc. Lucie Scholzová

Univerzita Pardubice  
Fakulta elektrotechniky a informatiky

Využití umělé inteligence pro správu skladů  
Diplomová práce

Univerzita Pardubice  
Fakulta elektrotechniky a informatiky  
Akademický rok: 2024/2025

# ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Bc. Lucie Scholzová**  
Osobní číslo: **I23282**  
Studijní program: **N0613A140007 Informační technologie**  
Téma práce: **Využití umělé inteligence pro správu skladů**  
Zadávací katedra: **Katedra softwarových technologií**

## Zásady pro vypracování

Diplomová práce "Využití umělé inteligence pro správu skladů" se zaměřuje na vývoj skladového systému řízeného chatbotem s umělou inteligencí. Aplikace, postavená na frameworku Django, umožňuje správu položek, uživatelů a klientů, stejně jako správu pozic, výdejek a příjmek. Aplikace bude nabízet dvě hlavní funkce: prvním je chatbot, který umožňuje základní správu systému, například vytváření a úpravu produktů a operací. Druhou funkcí je statistická analýza skladu, která pomocí předdefinovaných promptů generuje grafy a přehledy. Teoretická část práce se zabývá úvodem do Pythonu, skladovými operacemi a vývojem umělé inteligence.

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

Seznam doporučené literatury:

HORÁKOVÁ, Helena a KUBÁT, Jiří. *Řízení zásob: logistické pojetí, metody, aplikace, praktické úlohy*. 3., přeprac. vyd. Praha: Profess Consulting, 1999. ISBN 80-85235-55-2. DRAHOTSKÝ, Ivo a ŘEZNÍČEK, Bohumil. *Logistika: procesy a jejich řízení*. Brno: Computer Press, 2003. ISBN 80-7226-521-0. ENHOLM, Ida Merete; PAPAGIANNIDIS, Emmanouil; MIKALEF, Patrick a KROGSTIE, John. Artificial Intelligence and Business Value: a Literature Review. Online. *Information systems frontiers*. 2022, roč. 24, č. 5, s. 1709-1734. ISSN 1387-3326. Dostupné z: <https://doi.org/10.1007/s10796-021-10186-w>. [cit. 2024-09-26].

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

Datum zadání diplomové práce: **31. října 2024**  
Termín odevzdání diplomové práce: **23. května 2025**

**prof. Ing. Petr Doležel, Ph.D.** v.r.  
děkan

L.S.

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

V Pardubicích dne 29. listopadu 2024

Prohlašuji:

Práci s názvem Využití umělé inteligence pro správu skladů jsem vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila, jsou uvedeny v seznamu použité literatury.

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

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

V Pardubicích dne 20. 05. 2025

Bc. Lucie Scholzová

## ANOTACE

Diplomová práce „Využití umělé inteligence pro správu skladů“ se zaměřuje na návrh a vývoj webové aplikace StockWise, která využívá umělou inteligenci pro efektivní řízení skladových procesů. Systém je postaven na Django a Next.js, s databází Supabase. Klíčovými funkcemi jsou chatbot, který umožňuje správu položek a operací pomocí přirozeného jazyka, a statistický modul, který generuje přehledy a grafy na základě skladových dat. Teoretická část práce popisuje principy skladového hospodářství, umělé inteligence a použité technologie, zatímco praktická část se věnuje návrhu, implementaci a testování systému.

## KLÍČOVÁ SLOVA

skladové hospodářství, umělá inteligence, chatbot, webová aplikace, statistická analýza, Django, React, správa zásob

## TITLE

Utilization of Artificial Intelligence for Warehouse Management

## ANNOTATION

The thesis “Utilization of Artificial Intelligence for Warehouse Management” focuses on the design and development of the web application StockWise, which leverages artificial intelligence to improve warehouse operations. The system is built using Django and Next.js, with a Supabase database. Its key features include a chatbot that allows users to manage items and operations via natural language, and a statistics module that generates reports and visualizations based on warehouse data. The theoretical part covers warehouse management principles, artificial intelligence, and the technologies used, while the practical part focuses on system design, implementation, and testing.

## KEYWORDS

warehouse management, artificial intelligence, chatbot, web application, statistical analysis, Django, React, inventory control

# Obsah

Obsah .....	7
1. Úvod .....	12
2. Teoretická část .....	13
2.1. Základy skladového hospodářství.....	13
2.1.1. Definice a význam skladového hospodářství ve firemní logistice .....	13
2.1.2. Historický vývoj skladového hospodářství a jeho moderní přístupy.....	14
2.2. Sklad v logistickém řetězci .....	15
2.2.1. Typologie skladů .....	15
2.2.2. Klíčové procesy a operace .....	16
2.2.3. Typy zásob .....	17
2.2.4. Způsoby řízení zásob .....	17
2.2.5. Šarže, expirační datum, metody výdeje (FIFO, LIFO, FEFO).....	19
2.2.6. WMS systémy a jejich role .....	19
2.2.7. Automatická identifikace.....	20
2.3. Umělá inteligence a její vývoj .....	21
2.3.1. Základní principy AI a strojového učení .....	21
2.3.2. Historie AI a chatbotů.....	22
2.3.3. Jazykové modely (LLM), OpenAI a využití v praxi .....	23
2.3.4. Výhody a rizika použití AI ve firemních procesech.....	23
2.4. Přehled existujících řešení .....	23
2.4.1. Analýza aktuálních systémů využívajících AI pro správu skladů.....	24
2.5. Použité technologie a vývojové prostředí.....	25
2.5.1. Programovací jazyk Python .....	25
2.5.2. Webový framework Django.....	28
2.5.3. Knihovna React .....	30
2.5.4. Next.js – rozšíření Reactu pro vývoj webových aplikací .....	31
3. Experimentální část.....	33
3.1. Cílová skupina .....	33
3.2. Funkční požadavky .....	33
3.3. Nefunkční požadavky .....	34
3.4. Databázový návrh .....	34

3.4.1.	Analýza .....	35
3.4.2.	ERD .....	37
3.4.3.	Relační model dat .....	39
3.5.	Architektura systému .....	40
3.6.	Backendová část systému .....	41
3.6.1.	Struktura .....	42
3.6.2.	REST API .....	42
3.7.	Implementace chatbota .....	48
3.7.1.	Logika chatbota.....	48
3.8.	Frontendová část systému .....	51
3.8.1.	Struktura aplikace .....	51
3.8.2.	Uživatelské rozhraní .....	53
3.9.	Efektivita využití umělé inteligence.....	57
3.9.1.	Experiment 1 - výpis všech produktů .....	57
3.9.2.	Experiment 2 – nejjobsazenější pozice .....	58
3.9.3.	Pozorování z implementace a z experimentů .....	59
4.	Závěr .....	60
5.	Použitá literatura .....	61

## Seznam zdrojových kódů

Zdrojový kód 1 Python syntaxe Zdroj: vlastní .....	28
Zdrojový kód 2 ORM model v Djnago Zdroj: vlastní .....	29
Zdrojový kód 3 JSX v React Zdroj: vlastní .....	30
Zdrojový kód 4 State v React Zdroj: vlastní.....	30
Zdrojový kód 5 Struktura stránkované odpovědi Zdroj: vlastní.....	46
Zdrojový kód 6 Ukázka statistického promptu Zdroj: vlastní .....	50

## Seznam obrázků

Obrázek 1 Metoda ABC Zdroj: [20].....	18
Obrázek 2 Čárový kód Zdroj: [37] .....	20
Obrázek 3 QR kód Zdroj: [37].....	21
Obrázek 4 Proteus v Amazon Robotics Zdroj: [37] .....	24
Obrázek 5 ERD diagram Zdroj: vlastní .....	38
Obrázek 6 Relační model Zdroj: vlastní .....	39
Obrázek 7 Deployment diagram Zdroj: vlastní .....	41
Obrázek 8 Backend struktura aplikace StockWise Zdroj: vlastní .....	42
Obrázek 9 Frontend struktura aplikace StockWise Zdroj: vlastní.....	52
Obrázek 10 Tmavý a světlý mód Zdroj: vlastní .....	53
Obrázek 11 Stránka chatbota Zdroj: vlastní .....	56

## Seznam tabulek

Tabulka 1 Srovnání aktuálních integrací AI Zdroj: vlastní .....	25
Tabulka 3 Atributy serializeru ProductSerializer Zdroj: vlastní.....	45
Tabulka 4 Atributy serializeru ProductBulkSerializer Zdroj: vlastní.....	45
Tabulka 6 Ukázka odpovědí v prvním experimentu – výpis všech produktů Zdroj: vlastní....	58
Tabulka 7 Ukázka výsledku druhého experimentu – nejobsazenější pozice Zdroj: vlastní .....	59

## Terminologie

*SQL inection*: útok, při kterém útočník vkládá škodlivý SQL kód do vstupu, aby získal nebo upravil data v databázi. Bráníme se validací a parametrizovanými dotazy.

*Clickjacking*: podvodné přesměrování kliknutí na neviditelný prvek (např. tlačítko) s cílem oklamat uživatele. Ochrana např. pomocí hlavičky X-Frame-Options.

*Syntaxe*: pravidla pro správné psaní příkazů v jazyce (např. SQL nebo Python). Chybná syntaxe vede k chybám při spuštění.

*Konsolidace*: spojení více menších zásilek do jedné větší kvůli úspoře nákladů a efektivnějšímu doručení.

*Serializace*: proces převodu objektů (např. databázových záznamů) do formátu, který lze snadno přenášet (např. JSON) a naopak.

*Semiaktivní*: daný prvek není plně aktivní, ale vykonává určitou omezenou činnost nebo je připraven k aktivaci podle potřeby.

## Seznam zkratek

AI	Umělá inteligence
ML	Strojové učení
ORM	Objektově relační mapování (Object-Relational Mapper)
XSS	Cross-Site Scripting
CSRF	Cross-Site Request Forgery
CDN	Síť pro doručování obsahu (Content delivery network)
ERP	Plánování podnikových zdrojů
CRM	Řízení vztahů se zákazníky
RFID	Radio Frequency Identification
IoT	Internet věcí
WMS	Systém pro řízení skladů
LLM	Large Language Models
JWT	JSON Web Token
HR	Human resources

# 1. Úvod

V dnešní digitální době je efektivní správa skladových zásob jedním z klíčových faktorů ovlivňujících úspěšnost firem napříč odvětvími. Moderní podniky se neustále snaží zlepšit své logistické procesy, snížit provozní náklady a zvýšit přesnost při evidenci a manipulaci se zbožím. Skladování se tak stává nejen operativní činností, ale i strategickým prvkem celého dodavatelského řetězce.

Současně s tím, jak roste komplexnost skladových operací a požadavky na rychlost zpracování, se do popředí dostávají technologie umožňující jejich automatizaci. Mezi tyto technologie patří i umělá inteligence (AI), která nabízí možnosti zcela nového přístupu ke správě dat a analýze skladových procesů. Využití AI ve skladech umožňuje nejen přesnější předpovědi a rozhodování na základě historických dat, ale i snadnější interakci s uživatelem prostřednictvím tzv. inteligentních chatbotů.

Tato diplomová práce si klade za cíl navrhnout a implementovat jednoduchý skladový systém řízený chatbotem využívajícím umělou inteligenci, který bude schopen nejen zajišťovat základní operace, jako je příjem a výdej zboží, ale také poskytovat pokročilé statistické analýzy nad skladovými daty. Celý systém bude postaven na moderním webovém frameworku Django na straně backendu a Next.js na straně frontendu. Databázová vrstva bude řešena prostřednictvím Supabase. Cílem je rovněž zhodnotit efektivitu nasazení umělé inteligence ve formě chatbota a analyzovat, zda je jeho využití ve skladovém prostředí přínosné, přesné a prakticky využitelné.

Diplomová práce je rozdělena na dvě hlavní části, teoretickou, která se věnuje základům skladového hospodářství, popisu použitých technologií a principům umělé inteligence, a praktickou, která se zaměřuje na samotný návrh, vývoj a testování aplikace. Cílem je ukázat, jak lze kombinací moderních webových technologií a umělé inteligence vytvořit systém, který usnadní každodenní činnosti pracovníků skladu a zároveň poskytne vedení firmy užitečné informace pro rozhodování.

## 2. Teoretická část

### 2.1. Základy skladového hospodářství

Skladové hospodářství představuje klíčovou roli v podnikovém logistickém systému, jehož účelem je zajištění efektivního řízení materiálových toků, správu zásob a výrobních i distribučních procesů. Role skladů se v průběhu let výrazně proměnila od jednoduchého uskladnění zboží až po aktivní prvek, který přispívá k optimalizaci nákladů, zvyšování flexibility a rychlejší reakci na potřeby trhu.

Moderní skladové hospodářství neslouží pouze jako nástroj pro uchování zásob, ale také jako prostředek pro regulaci nerovnováhy mezi výrobou a spotřebou. Správně nastavené skladové procesy umožňují firmám předcházet výpadkům, optimalizovat pracovní tok a zajistit vysokou úroveň zákaznického servisu.

V této kapitole jsou shrnuty základní principy skladového hospodářství, jeho vývoj, význam v rámci firemní logistiky a hlavní cíle, které efektivní skladové řízení sleduje.

#### 2.1.1. Definice a význam skladového hospodářství ve firemní logistice

Skladové hospodářství představuje nedílnou součást podnikového logistického systému, jehož cílem je zajištění správného množství zboží, na správném místě, ve správný čas a za co nejnižší náklady. Někteří autoři, včetně Drahotského a Řezníčka [2], definují skladové hospodářství jako „soubor činností spojených s řízením, kontrolou, evidencí a pohybem zásob ve skladech“. Efektivní skladové hospodářství vytváří nezbytné podmínky pro stabilní a pružnou reakci firmy na proměnlivé požadavky trhu, čímž ovlivňuje její konkurenceschopnost.

Podle Horákové a Kubáta [1] je sklad jedním z regulačních článků logistického systému, který umožňuje kontinuitu provozu v situacích, kdy dochází k výkyvům v dodávkách či poptávce. V tomto smyslu plní sklady nejen funkci úložného prostoru, ale také funkci řízení toku materiálu v prostoru a čase.

Z pohledu firemní praxe má skladové hospodářství strategický význam. Nejde pouze o operativní činnost, ale o oblast, která může významně ovlivnit nákladovou strukturu podniku. Například náklady spojené s držetím zásob, manipulací, skladovým prostorem nebo investicemi do technologií mohou tvořit značnou část logistických výdajů firmy. Jak uvádí Chorpa a Meindl [3], správné řízení zásob a skladů může vést k optimalizaci provozních procesů a významně snížit celkové náklady v dodavatelském řetězci.

Důležitou roli hraje sklad také při zvyšování úrovně zákaznického servisu. Rychlá dostupnost požadovaného zboží, přesnost vychystávání, a minimalizace chyb při expedici jsou faktory, které přímo ovlivňují spokojenost zákazníků. V tomto kontextu je skladování vnímáno jako služba s přidanou hodnotou, nikoliv jako pouhý nákladový prvek [4].

S nástupem moderních technologií, automatizace a digitalizace se navíc skladové hospodářství stále více integruje s ostatními podnikovými systémy, zejména s ERP,

CRM nebo systémy řízení výroby. To přispívá k vyšší transparentnosti a přesnosti dat, a tím i k efektivnějšímu rozhodování na strategické i operativní úrovni.

Z výše uvedeného je patrné, že skladové hospodářství představuje klíčovou oblast firemní logistiky, která ovlivňuje nejen vnitřní fungování podniku, ale i jeho postavení v rámci celého dodavatelského řetězce.

### **2.1.2. Historický vývoj skladového hospodářství a jeho moderní přístupy**

Skladování jako činnost má velmi dlouhou historii, která sahá až do středověkých civilizací. Tehdy byly budovány sýpky a zásobárny pro uchování potravin a surovin v období přebytku. Z hlediska logistiky se však skladové hospodářství začalo formovat až v souvislosti s industrializací v 19. století, kdy se zvýšila potřeba centralizované správy materiálu, surovin a hotových výrobků [4].

Ve 20. století dochází k významnému posunu, zejména po druhé světové válce, kdy rychlý rozvoj průmyslové výroby a mezinárodního obchodu přinesl tlak na efektivní řízení zásob a přesné plánování logistických operací. Sklady se stávají nejen úložištěm zásob, ale také aktivním článkem při řízení výrobních a distribučních procesů. V tomto období vznikají první modely řízení zásob jako EOQ (Economic Order Quantity) nebo JIT (Just in Time), které zásadně ovlivňují podobu moderního skladového hospodářství [1].

V 70. a 80. letech 20. století dochází k první vlně automatizace skladových procesů, především ve formě mechanizovaných zakladačů a pásových dopravníků. Paralelně se začínají objevovat první skladové informační systémy (WMS), které umožňují elektronickou evidenci zásob a jejich pohybu. Významné je také zavádění čárových kódů, které zásadně zvýšily přesnost identifikace zboží a rychlost práce ve skladu.

Od počátku 21. století se skladové hospodářství stále více propojuje s celopodnikovými informačními systémy (ERP) a důraz se přesouvá na integraci dat, optimalizaci zásob, transparentnost zboží a rychlou reakci na poptávku zákazníka. Sklady se stávají „inteligentními“, čemuž napomáhá nasazení RFID, mobilních terminálů, automatických zakladačů řízených software, a nově i prvky strojového učení či prediktivní analýzy [3].

Současné trendy v oblasti skladového hospodářství odrážejí vývoj v oblasti digitalizace, automatizace a využití umělé inteligence. V moderním pojetí se již nejedná pouze o fyzický prostor pro uchování zboží, ale o strategické centrum datového řízení toků, které aktivně spolupracuje s ostatními složkami podniků i vnějším okolím. K tomu přispívá například využití robotických systémů pro vychystávání, například Amazon Robotics, hlasem řízené picking systémy, IoT senzory pro monitorování podmínek skladování, nebo chatboty a digitální asistenti zajišťující interakci člověk-systém v reálném čase.

Moderní skladové hospodářství tak reflektuje klíčové požadavky současné logistiky, jako jsou rychlost, přesnost, flexibilita a udržitelnost, a je zároveň podstatným faktorem pro budoucí rozvoj autonomních podnikových systémů.

## **2.2. Sklad v logistickém řetězci**

Sklad je základním článkem logistického řetězce, jenž zajišťuje plynulost toků materiálu, zboží a informací v rámci celého systému zásobování, výroby a distribuce. V moderní logistice přestává být vnímán pouze jako pasivní úložiště zásob a stává se aktivním prvkem řízení, plánování a optimalizace podnikových procesů. Sklad svým umístěním, vybavením a procesním nastavením přímo ovlivňuje jak rychlost a spolehlivost dodavatelského řetězce, ale také celkové náklady firmy a její flexibilitu vůči požadavkům trhu.

Podle Bowersoxe, Closse a Coopera [4] lze sklad považovat za kritický bod, kde se střetávají různé vstupy, jako jsou zboží od dodavatelů, požadavky z výroby a objednávky od zákazníků. Musí být řízen tak, aby nedocházelo ke zpoždění, ztrátám nebo neefektivní manipulaci. Efektivní řízení skladu zahrnuje jak fyzickou manipulaci se zbožím, tak informační a rozhodovací procesy, které umožňují správnou alokaci kapacit, plánování příjmů a výdejmů, a především koordinaci s dalšími logistickými články.

Z hlediska strategického plánování je důležité nejen typologické členění skladu, ale i jeho umístění, velikost, funkční členění a schopnost adaptace na měnící se provozní podmínky. Důraz je dnes kladen na flexibilitu, modularitu a možnost snadné škálovatelnosti.

V této kapitole budou představeny různé typy skladů a jejich specifické funkce, dále budou rozebrány klíčové procesy a operace, které ve skladech probíhají.

### **2.2.1. Typologie skladů**

Sklady lze rozlišovat podle celé řady hledisek, přičemž každé z nich reflektuje odlišnou dimenzi jejich provozní funkce. Mezi nejdůležitější patří zařazení skladu do výrobního nebo distribučního procesu. Na počátku výrobního cyklu se objevují tzv. vstupní sklady, které slouží k uchování surovin, komponent či materiálů určených pro výrobu. V rámci samotné výroby pak nacházejí uplatnění mezisklady, které vyrovnávají nerovnoměrnost mezi jednotlivými etapami výroby. Na jeho konci se nacházejí odbytové sklady, kde jsou uskladněny hotové výrobky připravené k expedici. [27]

Z hlediska prostorového uspořádání a obslužnosti lze rozlišit centralizované a decentralizované sklady. První jmenované představují hlavní konsolidační body, kde se soustřeďují zásoby pro celé podnikové nebo regionální území. Výhodou tohoto uspořádání je snadnější kontrola zásob a optimalizace nákupních procesů. Nevýhodou může být prodloužená dodací lhůta k zákazníkům. Oproti tomu decentralizované

sklady bývají umístěny blíže koncovým zákazníkům, což umožňuje rychlejší reakci na poptávku, ale zvyšuje náklady na řízení většího počtu distribučních bodů. [3]

Důležitou roli sehrávají distribuční sklady, které se neomezují pouze na skladování, ale často slouží jako místa pro doplňkové operace, jako je kompletace objednávek, etiketace či přebalování. Tyto sklady bývají umístěny strategicky blízko cílovým trhům a umožňují rychlou expedici zboží ke konečným zákazníkům. [26]

Specifickým případem je tzv. konsignační sklad, v němž je zboží fyzicky uloženo u odběratele, ale vlastnicky náleží dodavateli až do okamžiku jeho spotřeby. Tento model snižuje kapitálovou náročnost na straně odběratele a poskytuje větší flexibilitu v řízení zásob. Zároveň však klade vysoké nároky na vzájemnou důvěru mezi obchodními partnery a na přesné vymezení smluvních podmínek. [2]

Rostoucí oblibě se těší modely externí správy skladů, a to zejména ve formě tzv. logistiky třetí strany (3PL). V rámci tohoto přístupu přebírá provoz skladu specializovaný poskytovatel, který zabezpečuje nejen samotné skladování, ale i navazující logistické činnosti. Firmy tak získávají přístup k moderní infrastruktuře bez nutnosti vlastní investice a mohou se lépe soustředit na své klíčové aktivity. Další variantou je tzv. vendor-managed inventory (VMI), kdy dodavatel aktivně řídí zásoby, které jsou fyzicky uloženy u zákazníka. [28]

Kromě výše uvedených hledisek lze sklady dále třídit například podle umístění (vnitřní vs. vnější), úrovně ochrany před klimatickými vlivy (kryté vs. otevřené), či podle vlastnictví (vlastní vs. pronajaté). Některé sklady jsou úzce specializované, například příruční sklady, které slouží konkrétním pracovištím ve výrobě a umožňují rychlý přístup k často používaným položkám.

Z uvedeného vyplývá, že sklad není pouze pasivním úložištěm zboží, ale představuje aktivní prvek celého logistického systému. Jeho správné řízení a vhodné začlenění do podnikových procesů přispívá nejen k vyšší efektivitě operací, ale i k celkové konkurenceschopnosti podniku.

### **2.2.2. Klíčové procesy a operace**

Skladování hraje klíčovou roli v rámci celého logistického systému. Jeho hlavním cílem je efektivní a rychlé řízení toků materiálu od surovin přes polotovary až po hotové výrobky a zároveň zajištění přesných a aktuálních informací o stavu zásob. Mezi základní funkce skladování patří přesun produktů, jejich uskladnění a přenos informací o skladových položkách. [21]

Proces přesunu produktů zahrnuje několik klíčových operací. Příjem zboží zahrnuje vyložení zboží z dopravního prostředku, jeho vybalení, kontrolu fyzického stavu i porovnání s doprovodnou dokumentací. Následuje ukládání zboží, tedy přesun přijatých položek na určené místo ve skladu. Kompletace zboží pak znamená seskupení položek podle konkrétních objednávek zákazníků, často včetně přeskupení podle požadovaného sortimentu a množství. V některých případech dochází k překládce zboží přímo z příjmu do expedice bez meziskladu tzv. cross-docking.

Posledním krokem je odeslání zboží, tedy balení, značení a nakládka zásilek k dopravě.

Samotné uskladnění může mít různý charakter. Krátkodobé uskladnění slouží primárně pro plynulé doplňování zásob, zatímco dlouhodobější skladování pokrývá nadbytečné nebo sezónní zásoby mimo běžnou spotřebu.

Třetí důležitou složkou je přenos informací o zásobách, který musí být přesný a v reálném čase. Klíčové jsou údaje o aktuálním množství, stavu a umístění zásob, ale také o vstupech, výstupech, zákaznících a využití skladovací kapacity. Moderní sklady proto stále více využívají elektronické systémy přenosu dat, jako je EDI (Electronic Data Interchange), a technologie čárových kódů, které významně zvyšují rychlost i přesnost práce se skladovými informacemi. [21]

### **2.2.3. Typy zásob**

Zásoby představují hmotné statky, které jsou dočasně vyňaty z výrobního procesu nebo distribučního procesu s cílem zabezpečit plynulý chod podniku. V logistické praxi se zásoby obvykle člení podle funkce, kterou v systému plní. Výrobní zásoby zahrnují veškerý materiál nakoupený od dodavatelů, jako jsou suroviny a polotovary, a to od jejich pořízení až po vstup do výroby. Na ně navazují zásoby nedokončené výroby, tedy produkty, které už prošly některými výrobními operacemi, ale dosud nejsou hotové. Nakonec jsou zde zásoby hotových výrobků, které představují finální produkty připravené k dodání zákazníkům. [16]

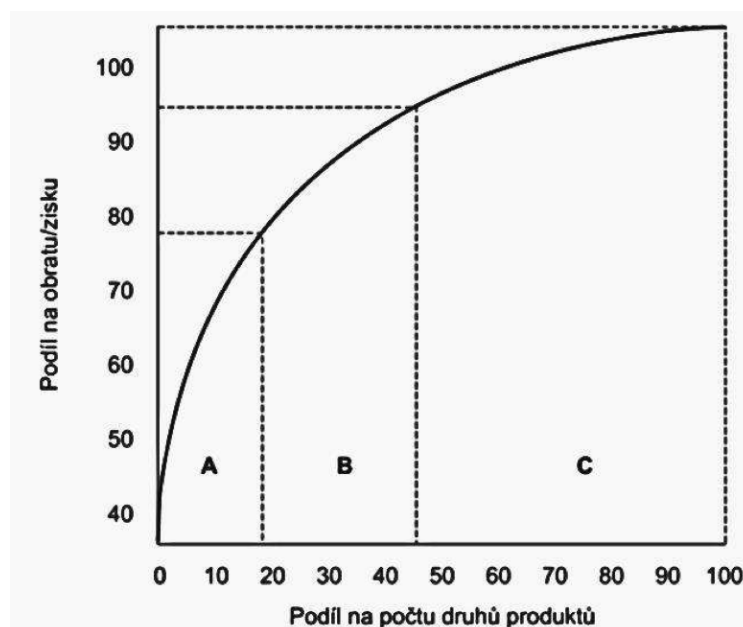
Dále můžeme zásoby dělit podle jejich funkce. Obrátová zásoba slouží k pokrytí spotřeby mezi dvěma dodávkami. Maximální a minimální zásoby pak představují hranice, mezi nimiž se množství zásob běžně pohybuje. Pojistná zásoba slouží k vyrovnání nepředvídatelných výkyvů v dodávkách i ve spotřebě materiálu nebo zboží. Vzniká zejména kvůli nejistotě při odhadu budoucího vývoje zásob a funguje jako ochrana v případech, kdy dodávka zboží nedorazí včas, nebo dorazí v menším množství, než se očekávalo. Technická zásoba pokrývá technologické požadavky, které je nutné splnit před použitím materiálu ve výrobě – například vysychání dřeva před jeho zpracováním při výrobě nábytku. Sezónní zásoba se vytváří tehdy, pokud lze určitou surovinu doplňovat jen v konkrétním období, nebo pokud je její spotřeba sezónní záležitostí. Zásoba se proto často doplňuje průběžně během celého roku. Spekulativní zásoba se udržuje s cílem využít výhodných nákupních podmínek a dosáhnout mimořádného zisku.

### **2.2.4. Způsoby řízení zásob**

Řízení zásob vyžaduje využití vhodných metod a modelů, které umožňují efektivně plánovat doplňování zásob, určovat optimální velikost objednávek i správné načasování jejich vystavení. Klíčovým cílem je sladění poptávky, nabídky a dodávek tak, aby nedocházelo k nadbytečným zásobám ani k jejich nedostatku.

Jedním z nejrozšířenějších přístupů je metoda Just-In-Time (JIT), která staví na úzké spolupráci mezi dodavatelem a odběratelem. Cílem je minimalizovat zásoby na straně odběratele tím, že materiál nebo zboží dorazí přesně ve chvíli, kdy je potřeba tedy v požadovaném množství, kvalitě a termínu. Tento přístup snižuje náklady na skladování a podporuje plynulost výrobních i logistických procesů.

Analýza ABC slouží k rozdělení zásob podle jejich významu z hlediska hodnoty. Základní myšlenkou je, že ne všechny položky mají stejnou finanční důležitost, malé procento položek obvykle tvoří většinu celkové hodnoty zásob. Zásoby se zde dělí do 3 skupin. Skupina A zahrnuje nejdůležitější položky, které tvoří 60–80% celkové hodnoty zásob, přestože představují 5–15 % položek. Skupina B obsahuje položky střední důležitosti, tvořící 15-25 % hodnoty a zhruba 15-25 % sortimentu. Nakonec skupina C zahrnuje většinu systému, ale s nízkým podílem na celkové hodnotě. Tato analýza je názorně ilustrována na následujícím grafu (viz obrázek níže), kde osa x reprezentuje podíl na počtu druhů produktů a osa y podíl na obrátu/zisku. [17]



Obrázek 1 Metoda ABC Zdroj: [20]

XYZ analýza slouží jako vhodné rozšíření k metodě ABC a zaměřuje se na stabilitu poptávky po jednotlivých položkách. I zde dochází k rozdělení sortimentu do tří skupin, tentokrát však podle pravidelnosti spotřeby. Položky ve skupině X mají velmi stabilní a předvídatelnou poptávku s minimálními výkyvy. Skupina Y zahrnuje položky s proměnlivou, ale ještě relativně předvídatelnou spotřebou. Naproti tomu skupina Z se vyznačuje velmi nepravidelným odběrem a těžko odhadnutelnou poptávkou.

Spojením přístupů ABC a XYZ lze získat komplexní pohled, který zohledňuje jak ekonomickou hodnotu zásob, tak i jejich spotřební charakter, což usnadňuje efektivní řízení zásob i plánování zásobovacích strategií.

### 2.2.5. Šarže, expirační datum, metody výdeje (FIFO, LIFO, FEFO)

V mnoha odvětvích, zejména potravinářství, farmaceutickém průmyslu, chemii či kosmetice, hrají šarže, expirační datum a metody výdeje klíčovou roli v oblasti řízení zásob. Správné sledování těchto údajů zajišťuje nejen soulad s legislativními požadavky a standardy kvality, ale i ochranu zákazníka a minimalizaci rizik spojenými s prošlým či neidentifikovatelným zbožím,

Šarže, označované také jako výrobní série nebo dávka, představuje určitou homogenní jednotku produkce, u níž lze předpokládat stejné vlastnosti, data výroby a případně i dobu použitelnosti. Každá šarže by měla být v systému jednoznačně identifikovatelná, aby bylo možné zpětně dohledat její pohyb napříč skladem i celým dodavatelským řetězcem. Evidence šarží je rovněž základem pro efektivní řízení stahování výrobků z trhu, což je důležité zejména u zboží podléhajícího přísným hygienickým a zdravotním normám.

Expirační datum, označované též jako doba použitelnosti nebo minimální trvanlivost, určuje časový horizont, během kterého lze daný produkt bezpečně a kvalitně použít. Ve WMS bývá expirační doba evidována společně s číslem šarže a lokalizací zboží. Při překročení stanoveného data by měl systém automaticky označit zboží jako neschopné výdeje a navrhnout jeho vyřazení, přepracování nebo likvidaci.

K efektivnímu řízení výdeje položek se v logistice uplatňují specifické strategie výdeje. Nejrozšířenějším je metoda FIFO (First In, First Out), která určuje, že jako první se vydává zboží, které bylo do skladu přijato nejdříve. Tato metoda je obzvláště výhodná pro zboží s omezenou trvanlivostí jako jsou například potraviny nebo rychle se kazící zboží.

V některých specifických případech se může uplatnit i metoda LIFO (Last In, First Out), který prioritně vydává zboží, které bylo přijato poslední. Tato metoda však v reálné praxi není běžná, neboť může vést k hromadění starších zásob ve skladu. LIFO se využívá zejména pro rozměrné a těžké zboží nebo sypký materiál, hlavně pro ulehčení manipulace tímto typem zboží.

Moderním přístupem, zejména u expiračního zboží, je metoda FEFO (First Expired, First Out), která zajišťuje výdej podle nejbližšího data expirace bez ohledu na datum přijetí. Tento přístup vyžaduje detailní sledování expirací a silné softwarové zázemí, ale přináší vysokou úroveň kontroly kvality a minimalizaci odpadu. FEFO je dnes běžnou praxí ve farmacii, potravinářství a dalších odvětvích, kde jsou přísné požadavky na bezpečnost a kvalitu zboží.

### 2.2.6. WMS systémy a jejich role

Systémy řízení skladu (WMS) představují klíčový nástroj pro digitalizaci a automatizaci skladových operací. Umožňují řídit celý proces od příjmu zboží, přes jeho uskladnění, až po expedici, a to s vysokou mírou přesnosti a efektivity.

Podle Gopalakrishnana a Haleema [22] je WMS softwarová aplikace určená pro centralizovanou správu zásob, řízení pohybu zboží ve skladu a koordinaci všech

činností spojených se skladovým hospodářstvím. Tyto systémy mohou fungovat buď jako samostatné aplikace, nebo být integrovány jako modul širšího ERP řešení.

Moderní WMS systémy často využívají technologie, jako je radiofrekvenční identifikace (RFID), hlasové ovládání, mobilní terminály či čárové kódy. Díky těmto prvkům lze výrazně zrychlit zpracování dat, minimalizovat chybovost a lépe sledovat aktuální stav zásob v reálném čase.

Hlavním přínosem WMS je zajištění plynulého toku zboží skladem při současném poskytování přesných a aktuálních informací managementu. Správně implementovaný systém výrazně zvyšuje produktivitu práce, zefektivňuje plánování a umožňuje flexibilní reakci na změny v poptávce nebo dodávkách. Automatizace rutinních činností zároveň umožňuje pracovníkům věnovat více času procesům s vyšší přidanou hodnotou, jako je například kontrola kvality nebo optimalizace skladových operací.

Díky tomu se WMS stávají nedílnou součástí moderního logistického řetězce a zásadním nástrojem pro firmy, které chtějí zůstat konkurenceschopné v podmínkách neustále se měnícího trhu.

### 2.2.7. Automatická identifikace

Automatická identifikace umožňuje rychlé a přesné sledování pohybu zboží, přičemž využívá různé technologie ke snímání dat bez nutnosti ručního zadávání. Mezi nejčastěji používané metody patří čárové kódy, RFID, EDI a QR kódy.

Čárové kódy patří mezi nejrozšířenější identifikační metody díky své jednoduchosti a nízké ceně. Zakódovaná data ve formě čar a mezer jsou snímána čtečkami a převáděna do číselné nebo textové podoby. Nejznámější formát je EAN (např. EAN-13), který identifikuje zemi, výrobce i konkrétní produkt. Výhodou čárových kódů je vysoká přesnost a rychlost čtení.



Obrázek 2 Čárový kód Zdroj: [37]

RFID (Radio Frequency Identification) je pokročilejší technologií, která umožňuje bezkontaktní identifikaci bez přímé viditelnosti. RFID tagy uchovávají data, která lze opakovaně číst i zapisovat pomocí čteček. Existují aktivní, pasivní a semiaktivní tagy, lišící se způsobem napájení. RFID zvyšuje efektivitu skladových procesů, umožňuje hromadné čtení a odolává náročným podmínkám. [18]

EDI (Electronic Data Interchange) slouží k elektronické výměně obchodních dokumentů mezi podniky. Umožňuje automatizované zaslání objednávek, faktur či dodacích listů ve standardizovaném formátu, čímž zrychluje komunikaci a snižuje chybovost i náklady spojené s papírovou agendou. [18]

QR kódy jsou dvourozměrné kódy schopné uchovávat větší množství informací. Nejčastěji se využívají v maloobchodu, marketingu, dopravě a platebních systémech. Uživatel je může snadno načíst pomocí mobilního zařízení a okamžitě získat přístup k potřebným údajům.



Obrázek 3 QR kód Zdroj: [37]

Tyto technologie se často kombinují, aby se dosáhlo co největší přesnosti, flexibility a automatizace skladových operací.

## **2.3. Umělá inteligence a její vývoj**

Umělá inteligence představuje rozsáhlý a dynamicky se rozvíjející obor informatiky, který se zaměřuje na vývoj systémů schopných vykonávat úkoly vyžadující lidskou inteligenci. Mezi klíčové oblasti patří rozpoznávání řeči, porozumění přirozenému jazyku, rozhodování na základě dat, samostatné učení a vizuální percepce. S prudkým nárůstem výpočetního výkonu, dostupností rozsáhlých datových souborů a pokroky v algoritmech došlo v posledním desetiletí k zásadnímu posunu od teoretických konceptů k praktickým aplikacím, které nacházejí uplatnění v řadě průmyslových i společenských oblastí.

### **2.3.1. Základní principy AI a strojového učení**

Strojové učení tvoří jeden z pilířů současné AI. Jeho základní princip spočívá v tom, že se algoritmus učí na základě dat bez explicitního naprogramování všech možných rozhodovacích pravidel. Na základě vstupních dat vytváří model, který je schopen predikovat nebo klasifikovat nové vstupy.

Z metod strojového učení lze jmenovat například logickou regresi, rozhodovací stromy, náhodné lesy, podpůrné vektorové stroje (SVM) a zejména neuronové sítě. Neuronové sítě, inspirované biologickou strukturou mozku, umožňují modelovat komplexní nelineární vztahy a nacházejí uplatnění zejména v oblastech jako je zpracování obrazu, přirozeného jazyka a zvuku. [29]

Rozšířením klasických neuronových sítí jsou hluboké neuronové sítě, které se skládají z více vrstev a umožňují učení na rozsáhlých datech s vyšší mírou abstrakce.

### 2.3.2. Historie AI a chatbotů

Historie umělé inteligence sahá až do poloviny 20. století. Již v roce 1956 byl na konferenci v Dartmoutu formálně položen základní rámec oboru. Mezi roky 1950 až 1960 došlo k položení prvních základních kamenů pro vývoj umělé inteligence. Alan Turing navrhl první test strojové inteligence, takzvaný Turingův Test, kde člověk musí rozpoznat, zda komunikuje s počítačem či člověkem. Byl také zahájen vývoj prvních průmyslových robotů. V navazujícím desetiletí byly uvedeny technologie, jakými byly například první chatovací programy jako Eliza, dále první programy hrající deskové hry na amatérské úrovni a první samostatně se pohybující roboti. Dalšími pokroky bylo zpracování přirozeného jazyka, první strojové překlady a zpracování obrazu. Na konci 70. let a taktéž na konci 80. let došlo kvůli nenaplněným očekáváním k útlumu počátečního optimismu ohledně umělé inteligence a následný pokles financování.

Strojové učení si opět získalo pozornost na začátku nového tisíciletí, a to především díky rostoucímu množství dostupných dat a zvyšujícímu se výpočetnímu výkonu. Naproti tomu neuronové sítě zůstávaly ještě nějakou dobu v pozadí. K výraznému rozvoji došlo až v období mezi lety 2010 a 2020, kdy se začaly trénovat hluboké neuronové sítě na rozsáhlých datových souborech, čímž vznikla samostatná oblast zvaná hluboké učení (deep learning). Díky tomu nastal průlom v oblastech, jako je rozpoznávání řeči, které dosáhlo úrovně srovnatelné s člověkem, mimo jiné díky nástupu hlasových asistentů, jako jsou Siri nebo Alexa. Významný pokrok nastal také v oblasti rozpoznávání obličejů na fotografiích. Umělá inteligence začala dosahovat nadlidského výkonu ve hrách, jako je například poker, a získala schopnost hrát jednoduché počítačové hry čistě na základě vizuálního vstupu. Mezi nejvýznamnější technologické milníky tohoto období pak patří vývoj autonomních vozidel schopných jízdy v běžném silničním provozu.

V posledních letech jsme svědky významného pokroku v oblasti umělé inteligence. Jedním z průlomů je využití neuronových sítí k určování trojrozměrné struktury proteinů, což ilustruje například projekt AlphaFold. Zároveň dochází k dynamickému rozvoji generativní umělé inteligence, která umožňuje automatizované vytváření obrazů, generování textů nebo tvorbu pokročilých chatovacích programů. S těmito technologickými skoky však roste i potřeba regulace, proto začínají vznikat první legislativní rámce, jako je například pakt Evropské unie, které se snaží nastavit pravidla pro bezpečné a etické využívání AI.

### **2.3.3. Jazykové modely (LLM), OpenAI a využití v praxi**

Velké jazykové modely (LLM) představují moderní přístup ke zpracování přirozeného jazyka. Jsou trénovány na miliardách slov pomocí architektur jako je Transformer, který umožňuje paralelní zpracování různých částí vstupního textu.

Mezi nejvýznamnější modely patří například GPT (Generative Pretrained Transformer) od společnosti OpenAI. Tento model je schopen generovat texty, překládat, sumarizovat, odpovídat na otázky či generovat kód. Praktické využití zahrnuje zákaznickou podporu, automatizaci tvorby dokumentů, personalizaci obsahu, asistenční nástroje nebo analýzu sentimentu.

Modely jako GPT-4 dosahují úrovně jazykového porozumění, které v mnoha ohledech konkuruje lidské schopnosti, i když stále trpí některými omezeními, jako je takzvaná halucinace, tedy vymýšlení si nepravdivých údajů. [31; 32]

### **2.3.4. Výhody a rizika použití AI ve firemních procesech**

Integrace umělé inteligence do podnikových procesů přináší firmám zásadní výhody, zejména v oblasti automatizace rutinních činností, prediktivní analýzy, optimalizace logistiky či personalizace zákaznické zkušenosti.

Například v oblasti HR lze pomocí AI efektivně filtrovat životopisy, předpovídat míru úspěšnosti kandidátů nebo analyzovat pracovní výkony. V marketingu umožňuje segmentaci zákazníků a cílené kampaně na základě behaviorálních vzorů. [33; 34]

Zároveň je však nutné reflektovat rizika jako je možnost zkreslení rozhodnutí způsobených nekvalitními trénovacími daty, ztrátu transparentnosti při využívání složitých modelů, takzvaný black-box problém, dále jsou zde etické otázky spojené se sledováním, sběrem a uchováváním dat a v neposlední řadě dopad na nezaměstnanost.

Z těchto důvodů je důležité přistupovat k nasazování AI s důrazem na etiketu, ochranu osobních údajů a právní regulaci.

## **2.4. Přehled existujících řešení**

S rychlým rozvojem umělé inteligence nachází její aplikace stále širší uplatnění v oblastech logistiky a skladového hospodářství. Firmy čelí vysokým nárokům na efektivitu, přesnost a rychlou reakci na měnící se tržní podmínky a obracejí se k pokročilým technologiím, které jim umožní optimalizovat své skladové procesy. Tato kapitola přináší přehled vybraných existujících řešení využívajících umělou inteligenci pro správu skladů. Zvlášť klade důraz na systémy jako SAP EWM, Amazon Robotics, Attobotics a Blue Yonder. Na základě analýzy jejich funkcí, architektur a reálných přínosů budou dále identifikovány jejich výhody a případná omezení.

### 2.4.1. Analýza aktuálních systémů využívajících AI pro správu skladů

SAP Extended Warehouse Management (EWM) je komplexní řešení, které integruje umělou inteligenci a strojové učení pro optimalizaci skladových operací. Systém umožňuje propojení skladových procesů s cloudovým ERP, čímž zajišťuje vyšší míru transparentnosti, predikci odchylek a inteligentní plánování kapacit. Umělá inteligence je v tomto případě využívána zejména pro prediktivní údržbu, plánování tras a optimalizaci skladových zásob. [35]

Amazon Robotics využívá sofistikované robotické systémy poháněné umělou inteligencí pro automatizaci ve svých distribučních centrech. Roboti jako Proteus využívají pokročilé metody počítačového vidění a senzorické technologie pro autonomní pohyb a interakci s okolním prostředím. Další technologie jako Sequoia slouží k prediktivnímu řízení skladování a vychystávání zboží. Tyto inovace vedly ke snížení nákladů na vyřízení objednávek o více než 25 %. [36]



Obrázek 4 Proteus v Amazon Robotics Zdroj: [37]

Attabotics představuje revoluční přístup k designu skladů inspirovaný strukturou mraveniště. Místo tradičních regálových systémů využívá 3D modulární konstrukci a roboty schopné pohybovat se jak horizontálně, tak vertikálně. Tímto způsobem lze dosáhnout až 85 % úspory prostoru a výrazného zrychlení objednávkových procesů. Klíčovým prvkem je software Weave, který optimalizuje tok materiálu v reálném čase.

Blue Yonder, dceřiná společnost Panasonicu, se specializuje na komplexní řešení dodavatelského řetězce. Jejich platforma kombinuje pokročilé algoritmy strojového učení s prediktivní analýzou pro optimalizaci zásob, přesnější plánování poptávky a podporu vícestupňové logistiky. Blue Yonder také podporuje omnichannel strategie, která usiluje o co nejefektivnější využití paralelních prodejních a komunikačních kanálů s cílem zkvalitnit celkový zákaznický zážitek, a flexibilní integraci s různými ERP systémy.

Tabulka 1 Srovnání aktuálních integrací AI Zdroj: vlastní

Systémy	Výhody	Nevýhody
SAP EWM	Komplexní integrace s ERP, prediktivní analýzy, flexibilní řízení procesů	Vyšší pořizovací náklady, delší doba implementace, potřeba školení
Amazon Robotics	Vysoká míra automatizace, snížení nákladů, zlepšení bezpečnosti práce	Vysoké počáteční investice, obavy s nahrazení lidské práce
Attabotics	Úspora prostoru, rychlé zpracování objednávek, modulární design	Nižší rozšíření na globálním trhu, omezené reference mimo Severní Ameriku
Blue Yonder	Pokročilé prediktivní funkce, integrace s různými platformami, podpora omnichannel strategií	Závislost na kvalitě dat, potřeba robustní IT infrastruktury

## 2.5. Použité technologie a vývojové prostředí

Při vývoji webových aplikací je důležité zvolit technologie, které umožňují rychlou práci s daty, vysokou flexibilitu a zároveň uživatelskou přívětivost. Mezi takové nástroje patří programovací jazyk Python, webový framework Django, knihovna React a vývojový framework Next.js. Každý z těchto nástrojů hraje důležitou roli při vývoji aplikací od serverové logiky po moderní interaktivní frontend.

Python je populární jazyk známý svou jednoduchostí, čitelností a širokou komunitou. Je vhodný jak pro začátečníky, tak pro profesionály. Nachází uplatnění v oblastech jako je vývoj webových aplikací, automatizace, datová analýza nebo strojové učení. Na pythonu je postaven framework Django, který poskytuje kompletní řešení tvorbu webových backendů, od správy databází až po zabezpečení a API.

Na straně klienta je využívána knihovna React, která umožňuje vytvářet rychlé a interaktivní uživatelské rozhraní prostřednictvím komponentového přístupu. Nad Reactem pak staví Next.js, který přináší rozšíření jako je server-side rendering, dynamické routování, optimalizaci výkonu a jednodušší deployment. Díky tomu je možné stavět moderní aplikace, které jsou zároveň rychlé, škálovatelné a dobře optimalizované i pro vyhledávače.

Tato kapitola se zaměřuje na představení těchto základních technologií.

### 2.5.1. Programovací jazyk Python

Programovací jazyk Python dnes patří mezi nejpoužívanější a nejvíce univerzální jazyky. Díky své jednoduché a přehledné syntaxi, vysoké výkonnosti a rozšiřitelnosti, si našel široké uplatnění. Python je vhodný nejen pro psaní jednoduchých scriptů,

ale také pro tvorbu rozsáhlejších nástrojů, desktopových aplikací, webových služeb, automatizací procesů a správou dat.

Významné zastoupení má Python v oblasti analýzy dat, strojového učení (ML) a umělé inteligence (AI), což z něj činí jeden z klíčových jazyků v dnešním výzkumu a vývoji v těchto oblastech. Díky projektům jako je MicroPython, je navíc možné Python využívat i pro vývoj aplikací běžících na mikrořadičích, a tedy i v oblasti embedded zařízení. I když je Python běžněji používán na straně serveru, existují také způsoby, jak jej nasadit i na straně klienta, například v rámci webového frontendu.

### **2.5.1.1. Historie a vývoj**

Počátky jazyka Python sahají do 80. let, kdy Guido van Rossum pracoval na jazyce ABC vyvinutém v nizozemském institutu CWI. ABC měl být jednoduchý a přívětivý jazyk pro neprogramátory, ale kvůli dominanci jazyka BASIC se příliš nerozšířil. Přesto se stal inspirací pro Python, zejména svou jednoduchou syntaxí. [9]

Guido van Rossum chtěl zachovat výhody ABC, ale odstranit jeho omezení (např. nemožnost rozšíření o moduly), a tak v roce 1991 vydal první verzi Pythonu. Ta nabízela jednoduchou syntaxi, automatickou správu paměti a podporu pro moduly a výjimky.

V roce 2000 přišel Python 2 s podporou Unicode a list comprehensions. Kvůli problémům se zpětnou kompatibilitou se však obtížně udržoval, což vedlo k vývoji Pythonu 3, vydaného v roce 2008. Ten přinesl výrazné změny (např. print jako funkci, sjednocení práce s textem a bajty), ale adopce trvala několik let. Oficiální podpora verze 2 skončila v roce 2020.

Od verze 3.6 se Python stal plně moderním jazykem, hojně využívaným ve webovém vývoji, datové analýze, AI nebo IoT. Přesto má i své nevýhody, např. nižší výkon oproti kompilovaným jazykům nebo komplikace při správě typů ve větších projektech. [9]

Python si zachoval svou jednoduchost, ale vyrostl v jeden z nejuniverzálnějších jazyků současnosti, jehož vývoj směřuje k vyšší výkonnosti, bezpečnosti a moderním standardům.

### **2.5.1.2. Syntaxe**

Jedním z hlavních důvodů popularity Pythonu je jeho jednoduchá, přehledná a srozumitelná syntaxe. Na rozdíl od tradičních programovacích jazyků, jako jsou C, Java nebo C#, které pro vymezení bloků kódu používají složené závorky, Python sází na povinné odsazení. To znamená, že struktura programu je přímo určena tím, jak jsou jednotlivé řádky odsazené, nesprávná nebo nekonzistentní indentace proto vede k chybě. Tento přístup zvyšuje čitelnost a zároveň nutí programátora psát přehledněji. [10; 11]

Další odlišností je způsob práce s proměnnými. Python je dynamicky typovaný jazyk, takže přiřazení hodnoty automaticky určuje její datový typ. Není tedy nutné předem deklarovat, že proměnná *x* je celé číslo nebo řetězec, stačí jí jednoduše přiřadit hodnotu. Díky tomu je kód stručnější a flexibilnější, což výrazně urychluje vývoj.

Syntaxe podmínkových výrazů a cyklů je v Pythonu velmi úsporná a čitelná. Není potřeba používat závorky kolem logických výrazů, jak je běžné v jiných jazycích. Rozhodující je klíčové slovo (např. *if*, *for*, *while*) a dvojtečka na konci řádku. Samotná těla podmínek či cyklů jsou pak odsazena, jak bylo zmíněno výše. Například jednoduchá podmínka nebo iterace přes seznam tak v Pythonu zabírá minimum řádků, a přitom zůstává přehledná.

Další silnou stránkou Pythonu je práce s funkcemi. Ty se definují pomocí klíčového slova *def* a kromě běžných parametrů mohou využívat výchozí hodnoty, pojmenované argumenty nebo dokonce proměnný počet parametrů pomocí *\*args* a *\*\*kwargs*. Od verze 3.5 je navíc možné přidávat typové anotace, což zvyšuje srozumitelnost kódu a umožňuje jeho kontrolu pomocí statických nástrojů. [9;10; 11]

Python se také vyznačuje bohatou sadou vestavěných datových struktur, například seznamy (*list*), slovníky (*dictionary*), množiny (*set*) nebo n-tice (*tuple*), které jsou díky jednoduché syntaxi snadno použitelné i pro začátečníky. Tyto struktury jsou nejen snadno čitelné, ale i výkonné a dobře integrované do jazyka.

Modularita je další oblastí, kde Python vyniká jednoduchostí, každý soubor s příponou *.py* je modul, který lze importovat do jiného skriptu pomocí klíčového slova *import*. Kromě toho Python podporuje standardní knihovny i externí balíčky, které se snadno instalují pomocí nástroje *pip*. [10; 11]

Celkově lze říci, že syntaxe Pythonu je navržena tak, aby byl kód co nejčistší a nejlépe čitelný, což ho činí ideálním jazykem nejen pro začátečníky, ale i pro rychlý vývoj složitějších aplikací.

```
# Import modulu
import math

# Funkce pro výpočet odmocniny, pokud je číslo kladné
def zpracuj_cislo(x: float) -> str:
    if x < 0:
        return "Záporné číslo – nelze odmocnit"
    elif x == 0:
        return "Nula"
    else:
        odmocnina = math.sqrt(x)
        return f"Odmocnina z {x} je {odmocnina:.2f}"

# Seznam čísel k vyhodnocení
cisla = [16, 0, -4, 9]

# Iterace přes seznam
```

```
for cislo in cisla:
    vysledek = zpracuj_cislo(cislo)
    print(vysledek)
```

*Zdrojový kód 1 Python syntaxe Zdroj: vlastní*

### 2.5.1.3. Využití v různých oblastech

Python je univerzální jazyk s širokým uplatněním díky jednoduché syntaxi a bohatému ekosystému knihoven. Ve webovém vývoji se používá díky frameworku jako Django, Flask či FastAPI. [13] Ve vědě nebo výzkumu slouží k numerickým výpočtům, vizualizacím a simulacím pomocí knihoven jako NumPy, SciPy nebo Matplotlib. V oblasti datové vědy je standardem pro analýzu a vizualizaci dat díky nástrojům jako je Pandas, Jupyter a Plotly. V umělé inteligenci a strojovém učení dominuje díky knihovnám jako TensorFlow, PyTorch či Scikit-learn. Dále je často využíván pro automatizaci úloj, skriptování a práci s API. [9; 10]

## 2.5.2. Webový framework Django

Django je open-source webový framework pro jazyk Python, navržený pro rychlý a bezpečný vývoj webových aplikací. Jeho hlavní filozofií je „batteries included“, tedy přístup, kdy vývojáři dostávají základní množství nástrojů a knihoven potřebných pro vývoj bez nutnosti stahovat externí moduly.

Django vznikl v redakčním prostředí amerického zpravodajství (Lawrence Journal-World) jako interní nástroj pro rychlou tvorbu obsahově náročných aplikací. Dnes patří k nejpopulárnějším Python frameworkům a je široce využíván v komerční i nekomerční sféře.

### 2.5.2.1. Architektura frameworku (MVT model)

Django je postaveno na architektuře Model-View-Template (MVT), která je podobná známému vzoru MVC (Model-View-Controller), ale s vlastním názvoslovím. Model zajišťuje datovou vrstvu aplikace. Definují se pomocí Python tříd, které reprezentují databázové tabulky. Django automaticky generuje databázové schéma z těchto tříd. View obsahuje logiku pro zpracování požadavků a vrací odpovědi, například HTML, JSON, XML. V Djangovém pojetí neznámá „view“ uživatelské rozhraní, ale spíše řídicí logiku odpovědi. Nakonec Template představuje prezentační vrstvu, HTML šablony s vlastním šablonovacím jazykem, do kterých se vkládají data z view.

Tato architektura zajišťuje čisté oddělení zodpovědnosti jednotlivých částí aplikace, podporuje škálovatelnost a zjednodušuje údržbu.

### 2.5.2.2. Hlavní vlastnosti a výhody

Django je vysoce komplexní a robustní webový framework, který nabízí celou řadu vestavěných modulů a nástrojů. Ty umožňují vývojářům soustředit se především na byznys logiku aplikace namísto implementace základních funkcí. Mezi klíčové vlastnosti patří ORM (Object-Relational Mapping), nástroj pro práci s databází pomocí Python objektů bez nutnosti psát SQL dotazy, vestavěná administrace, která umožňuje automaticky generované rozhraní pro správu dat podle definovaných modelů. Dalšími vlastnostmi je například vlastní autentizační systém, možnost překlada aplikace do různých jazyků (Mezinárodní lokalizace (i18n)) a integrovaný testovací framework (unittest) díky kterému je umožněno testování aplikace přímo v rámci frameworku.

Použití frameworku Django přináší vývojářům několik zásadních výhod. Jednou z nich je rychlý vývoj, který umožňuje tvorbu plně funkčních aplikací ve velmi krátkém čase díky rozsáhlé sadě vestavěných nástrojů, včetně administračního rozhraní a autentizačního systému. Dalším významným přínosem je bezpečnost. Django je navrženo s ohledem na ochranu před nejčastějšími webovými hrozbami, jako jsou XSS, CSRF, SQL injection nebo clickjacking, a poskytuje bezpečné výchozí nastavení již po instalaci. Modularita frameworku umožňuje rozdělit aplikaci do menších komponent (apps), které lze samostatně vyvíjet, opakovaně používat a snadno udržovat, což přispívá k přehlednosti i škálovatelnosti větších projektů. Právě škálovatelnost je dalším důvodem, proč je Django vhodné i pro rozsáhlé a zatěžované aplikace, nabízí podporu pro cachování, middleware vrstvu a bezproblémovou integraci s moderními technologiemi, jako jsou frontendové frameworky typu React či Vue, nebo CDN pro optimalizované doručování obsahu.

Díky těmto vlastnostem je Django vhodnou volbou jak pro menší projekty, tak i pro rozsáhlé produkční aplikace s vysokými nároky na výkon, bezpečnost a rozšiřitelnost.

### 2.5.2.3. Práce s databázemi a ORM

Jak již bylo zmíněno v předešlé kapitole, jedním ze stěžejních rysů Django je jeho vlastní ORM. Umožňuje pracovat s relační databází pomocí Python kódu bez nutnosti psát SQL dotazy. Vývojář definuje model jako běžnou python třídu, přičemž každé pole třídy reprezentuje sloupec v databázové tabulce.

```
from django.db import models

class Product(models.Model):
    sku = models.CharField(max_length=100, unique=True)
    name = models.CharField(max_length=255)
    description = models.TextField(blank=True)
    created_at = models.DateTimeField('Created At', auto_now_add=True)
    client = models.ForeignKey('client.Client', verbose_name='Client', null=False,
on_delete=models.CASCADE)
    amount_cached = models.IntegerField(default=0)
```

Zdrojový kód 2 ORM model v Django Zdroj: vlastní

Po vytvoření modelů lze pomocí příkazu *makemigrations* a *migrate* automaticky vygenerovat databázové schéma. ORM podporuje běžné dotazy jako je *filter()*, *get()*, *exclude()*, *annotate()* a *aggregate()*, ale v případě potřeby umožňuje i použití nativního SQL.

Django je kompatibilní s několika databázemi jako jsou PostgreSQL, MySQL, SQLite nebo Oracle.

### 2.5.3. Knihovna React

React je moderní JavaScriptová knihovna pro tvorbu UI, vyvinutá společností Meta (dříve Facebook). První verze Reactu byla představena v roce 2013 a od té doby se stala jedním z nejpobulárnějších nástrojů pro vývoj interaktivních a reaktivních webových aplikací. Je zaměřena především na tvorbu front-endu, přičemž díky jejímu komponentovému přístupu lze efektivně řídit logiku a vzhled uživatelského rozhraní.

#### 2.5.3.1. Základní principy Reactu (komponenty, JSX, props, state)

React staví na několika klíčových konceptech, které umožňují efektivní vývoj. Vše je tvořeno pomocí komponent neboli samostatných stavebních bloků, které zapouzdřují logiku, strukturu a styl určité části uživatelského rozhraní. Komponenty mohou být funkcionální (běžnější přístup v moderním Reactu) nebo třídové.

JSX, JavaScript XML, je rozšířená syntaxe JavaScriptu umožňující zapisovat HTML podobný kód přímo do JavaScriptových souborů. JSX zvyšuje čitelnost a propojuje logiku s vykreslováním.

```
const Pozdrav = (props) => <h1>Ahoj, {props.jmeno}!</h1>;
```

Zdrojový kód 3 JSX v React Zdroj: vlastní

Props, neboli vlastnosti, představují způsob, jak předávat data komponentám. Jsou neměnné, tedy read-only, a slouží ke konfiguraci komponenty zvenčí.

State představuje vnitřní data komponenty, které se mění. Při změně stavu dochází k opětovnému vykreslení komponenty. V moderním Reactu se ke správě stavu často používá hoko `useState`.

```
const Pocitadlo = () => {  
  const [pocet, setPocet] = useState(0);  
  return <button onClick={() => setPocet(pocet + 1)}>Klikni: {pocet}</button>;  
};
```

Zdrojový kód 4 State v React Zdroj: vlastní

### 2.5.3.2. Virtuální DOM a efektivita Reactu

Jedním z hlavních důvodů, proč je React velmi rychlý a efektivní, je použití virtuálních DOMu. Tradiční manipulace s DOM stromem v prohlížeči je pomalá, protože každý zásah může způsobit přepočítání a překreslení celého rozhraní.

React tuto situaci řeší tak, že udržuje virtuální reprezentaci DOMu v paměti. Při změně stavu komponenty React nejprve provede změnu v této virtuální kopii a následně pomocí algoritmu porovnání provede pouze nezbytné aktualizace ve skutečném DOMu.

Tento přístup výrazně zvyšuje výkon aplikace, snižuje počet zbytečných operací a činí React vhodným i pro náročné interaktivní rozhraní.

### 2.5.4. Next.js – rozšíření Reactu pro vývoj webových aplikací

Next.js je moderní framework postavený na knihovně React, který přináší nadstavbu pro vývoj produkčně nasaditelných webových aplikací. Vyvíjí jej společnost Vercel a jeho cílem je zjednodušit tvorbu uživatelského rozhraní. Poskytuje ucelený nástroj pro tvorbu celého frontendu včetně routování, vykreslování stránek na straně serveru, optimalizace výkonu, a dokonce i definice jednoduchého API.

Jedním z hlavních přínosů Next.js je takzvaný souborový routing, který umožňuje definovat jednotlivé stránky aplikace jednoduše pomocí souborů ve složce /pages. Každý soubor zde reprezentuje jednu veřejně dostupnou stránku a odpovídá konkrétní URL adrese, čímž se výrazně ulehčuje práce s routováním bez nutnosti ruční konfigurace.

Další klíčovou vlastností je podpora různých strategií vykreslování, kde si vývojář může sám zvolit mezi server-side renderingem (SSR), statickým generováním (SSG) nebo hybridním přístupem zvaným incremental static regeneration (ISR). Díky tomuto lze snadno optimalizovat načítání obsahu s ohledem na SEO, rychlost a aktuálnost dat. SSR je výhodný například pro dynamický obsah nebo přihlášené uživatele, zatímco SSG se hodí pro statické stránky a blogy. ISR umožňuje generovat statické stránky při buildu, ale zároveň je později automaticky obnovovat podle potřeby.

Velkou výhodou je také možnost definovat API endpointy přímo v rámci frontendového projektu. Ve složce /pages/api lze vytvořit jednoduché backendové funkce, které zpracovávají http požadavky jako jsou například formuláře, dotazy na databázi či přístup k externím službám. Tento přístup výrazně zjednodušuje architekturu menších aplikací a umožňuje vyvíjet fullstack aplikace bez potřeby samostatného serveru.

Next.js dále nabízí řadu funkcí pro optimalizaci výkonu, včetně inteligentního načítání obrázků pomocí komponenty <Image>, automatického rozdělování kódu podle stránek a podporu lazy-loadingu. Podporuje také TypeScript, CSS moduly, Tailwind CSS a další moderní nástroje a knihovny běžně používané v React ekosystému.

Mezi nevýhody lze zařadit vyšší komplexitu oproti čistému Reactu. Vývoj vyžaduje porozumění jak serverové, tak klientské části aplikace a práci s různými metodami

načítání dat. Dále je nutné pochopit, jak framework generuje jednotlivé stránky. I přes to, je však pro většinu projektů výhodnější volbou než čistý React, zejména pokud je cílem vytvořit produkčně připravený web s důrazem na výkon, SEO a správu dat.

Next.js je dnes využíván mnoha významnými firmami a platformami. Mezi ně patří například TikTok, Twitch, Hulu, Nike, Vercel.com a Notino. Jeho popularita stále roste, zejména díky jednoduchému nasazení, aktivní komunitě a dobré dokumentaci.

### **3. Experimentální část**

Praktická část této diplomové práce se věnuje realizaci komplexní webové aplikace určené pro efektivní správu skladového hospodářství. Aplikace je navržena tak, aby odpovídala aktuálním požadavkům na digitalizaci skladových procesů, a zároveň umožňovala využití prvků umělé inteligence pro usnadnění interakce uživatelů se systémem. Důraz je kladen na uživatelskou přívětivost, bezpečnost, přehlednost a automatizaci rutinních operací.

V jednotlivých podkapitolách je popsán postupný vývoj řešení, od počáteční analýzy a návrhu, přes implementaci jádra systému a jeho jednotlivých komponent, až po testování a vyhodnocení. V závěru je reflektováno naplnění cílů a zhodnocen celkový přínos aplikace v kontextu reálného použití.

#### **3.1. Cílová skupina**

Aplikace je určena především pro pracovníky ve skladech, správce skladového hospodářství a menší až středně velké firmy, které hledají efektivní nástroj pro evidenci a správu svých zásob. Díky uživatelské přívětivosti a využití moderních technologií, jako je umělá inteligence, je systém vhodný jak pro firmy bez zavedeného informačního systému, tak pro organizace, které chtějí nahradit zastaralé nebo nevyhovující řešení modernější alternativou.

Primární uživatelskou skupinou jsou skladníci, kteří aplikaci využívají pro každodenní operace, jako je příjem a výdej zboží, kontrola stavů zásob, nebo manipulace s produkty. Tito uživatelé ocení jednoduché, rychlé a intuitivní rozhraní optimalizované pro použití na běžných počítačích, tabletech nebo mobilních zařízeních.

Sekundární cílovou skupinou jsou vedoucí pracovníci a správci skladu, kteří aplikaci využívají ke sledování statistik, reportingu a strategickému rozhodování. Pro tyto uživatele je klíčová možnost přehledného zobrazení dat, statistiky a predikce vytvořených umělou inteligencí.

Díky integraci chatbota založeného na umělé inteligenci je systém zároveň vhodný i pro méně technicky zdatné uživatele, kteří mohou se skladem komunikovat prostřednictvím přirozeného jazyka bez potřeby složité orientace v rozhraní.

#### **3.2. Funkční požadavky**

Aplikace StockWise je navržena jako komplexní nástroj pro správu skladového hospodářství s důrazem na efektivitu, přehlednost a podporu ze strany umělé inteligence. Z toho vyplývá sada funkčních požadavků, které definují chování systému a rozsah jeho základních funkcí:

- FR 01. Aplikace bude umožňovat registraci uživatele
- FR 02. Aplikace bude umožňovat přihlášení pomocí e-mailu a hesla
- FR 03. Aplikace bude umožňovat změnu hesla, jména a osobních údajů
- FR 04. Aplikace bude umožňovat přidání nového produktu
- FR 05. Aplikace bude umožňovat úpravu a smazání produktu
- FR 06. Aplikace bude umožňovat sledování šarží, jejich množství a expirace
- FR 07. Aplikace bude umožňovat vyhledávání a filtrování dat podle parametrů
- FR 08. Aplikace bude umožňovat provádění skladových operací (příjem, výdej)
- FR 09. Aplikace bude uchovávat historii všech skladových operací
- FR 10. Aplikace bude umožňovat definici pozic ve skladu a přiřazení produktů
- FR 11. Aplikace bude umožňovat ukládání produktů do krabic a sledování jejich obsahu
- FR 12. Chatbot bude umožňovat kladení dotazů a zadávání jednoduchých pokynů
- FR 13. Chatbot bude umožňovat nahrávání dat ve formátu CSV
- FR 14. Chatbot bude generovat statistiky
- FR 15. Aplikace bude zobrazovat vizualizace jako grafy a tabulky

### 3.3. Nefunkční požadavky

Nefunkční požadavky specifikují kvalitativní vlastnosti systému, které se netýkají přímo jeho funkčnosti, ale zásadním způsobem ovlivňují použitelnost, výkonnost, bezpečnost a udržitelnost aplikace. V případě aplikace StockWise jsou klíčové zejména následující aspekty:

- FR 01. Aplikace bude dostupná ve formě webové aplikace
- FR 02. Aplikace bude dostupná 24/7 při nasazení do produkčního prostředí
- FR 03. Aplikace bude podporovat aktuální verze prohlížečů
- FR 04. Frontend bude napsán v Reactu a frameworku Next.js
- FR 05. Backend bude napsán v Django
- FR 06. Aplikace bude využívat Supabase cloudovou databázi
- FR 07. Aplikace bude umožňovat horizontální škálování backendu i frontendu
- FR 08. Aplikace bude mít intuitivní uživatelské rozhraní a být použitelná i na mobilních zařízeních
- FR 09. Aplikace bude umožňovat snadné rozšíření o nové funkce
- FR 10. Aplikace bude vyžadovat staré heslo při změně hesla uživatele
- FR 11. Aplikace bude umožňovat reset hesla pouze s platným tokenem
- FR 12. Aplikace bude dostupná v tmavém i světlém režimu
- FR 13. Backend bude oddělen od frontendu pro snadné nasazení a údržbu
- FR 14. Backend bude poskytovat API dokumentaci (např. pomocí Swagger)

### 3.4. Databázový návrh

Databázová struktura navržená v rámci této aplikace vychází z požadavků na správu více skladových jednotek, přehledné evidování produktů včetně šarží, sledování jejich

pohybu a historie, a také z potřeby podpory uživatelského rozhraní s přizpůsobitelnými dashboardy. Celý návrh reflektuje specifika systému zaměřeného na operativní skladové procesy, správu klientů a jejich položek, přebalování zboží, pohyb obalů po skladových pozicích a historii všech důležitých změn v systému.

Hlavním cílem bylo vytvořit systém, který bude plně připraven pro nasazení v praxi s možností dalšího rozšiřování. Návrh proto počítá s více klienty (multitenantní struktura), více sklady, flexibilní vazbou mezi produkty a jejich šaržemi, a také s důslednou evidencí operací ve skladu. Důraz je kladen na jednoznačnou identifikaci všech objektů v systému a na integritu dat napříč celou databází.

### 3.4.1. Analýza

#### 3.4.1.1. Scénář

**Uživatel** si chce vytvořit profil. Nejprve musí vyplnit uživatelský email, kterým se bude následně přihlašovat a heslo. Po registraci se uživatel může přihlásit. Uživatel může změnit heslo.

Uživatel chce vložit nového **klienta**. Nejprve musí zadat název klienta, jeho identifikační údaje a vytvořit tím záznam v databázi. Klientovi je automaticky přiřazen unikátní identifikátor, pod kterým budou dále evidována všechna jeho data. Pokud chce uživatel daného klienta spravovat, musí ho mít v seznamu spravovaných klientů.

Uživatel chce vytvořit nový **sklad**. Nejdříve musí zadat název skladu a popis. Každý sklad je dále rozdělen na **pozice**, které je třeba vytvořit. Při vytváření pozice je nutné zadat její kód a přiřadit ji ke konkrétnímu skladu.

Pracovník skladu chce vložit do systému nový **produkt**. Nejprve musí zadat název produktu, jeho SKU kód a další technické parametry. Produkt musí být přiřazen ke konkrétnímu klientovi.

Pracovník chce zaevidovat novou **krabici**. Nejprve vyplní její EAN, šířku, výšku, hloubku a váhu a přiřadí ji na odpovídající pozici.

Pracovník skladu chce vytvořit **šarži** k produktu. Při vytváření šarže je nutné zadat název šarže a datum expirace. Každá šarže je jednoznačně navázána na produkt.

Skladník chce zadat novou skladovou **operaci**. Nejprve vybere typ operace (příjem, výdej). Poté do operace přidá jednotlivé položky. U každého produktu je třeba zadat množství, a volitelně šarži a balení, do kterého má být položka umístěna.

Každý provedený krok v rámci operací, produktů, šarží, skupin a pozic je zaznamenán do **historie**. Záznam do historie obsahuje čas, typ události, uživatele, který akci provedl a odkaz na související entitu (například skupinu, balení nebo pozici).

Uživatel si chce upravit svůj **dashboard**. V systému vytvoří nebo upraví konfiguraci panelů dashboardu, která se ukládá do databáze společně s jeho uživatelským identifikátorem a typem dashboardu (např. hlavní přehled nebo statistiky).

### 3.4.1.2. Strukturální pravidla

Uživatel musí obsahovat e-mail, heslo a identifikátor profilu. Uživatel může mít přiřazenou konfiguraci dashboardu a může spravovat jednoho nebo více klientů. Klient musí obsahovat název, IČ, DIČ a unikátní identifikátor. Každý klient může být spravován jedním nebo více uživateli. Uživatel může spravovat jednoho nebo více klientů. Sklad musí obsahovat název, popis a id klienta, kterému patří. Každý sklad může obsahovat jednu nebo více pozic. Pozice musí obsahovat unikátní kód a id skladu, ke kterému náleží. Každá pozice může být přiřazena jedné nebo více krabicím. Produkt musí obsahovat název, SKU, technické parametry (např. rozměry, hmotnost) a id klienta, kterému produkt náleží. Každý produkt může mít více šarží. Krabice (balení) musí obsahovat EAN, šířku, výšku, hloubku, váhu a id pozice, na které se nachází. Šarže musí obsahovat název, datum expirace a id produktu, ke kterému náleží. Každá šarže může být použita v jedné nebo více skladových operacích. Skladová operace musí obsahovat typ operace (např. příjem, výdej), datum vytvoření, id uživatele a seznam skupin. Každá skupina musí obsahovat množství a volitelně id šarže, id krabice a id operace. Historie musí obsahovat čas akce, typ události, id uživatele, který akci provedl, a id související entity (například pozice, balení, operace nebo produkt).

### 3.4.1.3. Procedurální pravidla

Před zadáním klienta do databáze musí být nejdříve vytvořen záznam v tabulce user\_user, pokud má být klient přiřazen spravujícímu uživateli. Před přiřazením klienta uživateli musí být vytvořen záznam v tabulkách client\_client a user\_user. Před zadáním skladu musí být nejdříve vytvořen záznam v tabulce client\_client. Před zadáním pozice musí být nejdříve vytvořen záznam v tabulce warehouse\_warehouse. Před zadáním produktu musí být nejdříve vytvořen záznam v tabulce client\_client. Před zadáním šarže musí být nejdříve vytvořen záznam v tabulce product\_product. Před zadáním krabice (balení) musí být nejdříve vytvořen záznam v tabulce position\_position. Před zadáním skladové operace musí být nejdříve vytvořen záznam v tabulce user\_user. Před zadáním položky do skladové operace musí být nejdříve vytvořen záznam v tabulkách operation\_operation, product\_product a volitelně batch\_batch a box\_box. Před zadáním záznamu do historie musí být nejdříve vytvořen záznam v tabulce, ke které se vztahuje (například operation\_operation, product\_product, box\_box, position\_position nebo group\_group). Před zadáním konfigurace dashboardu musí být nejdříve vytvořen záznam v tabulce user\_user.

### 3.4.1.4. Analýza

Uživatel obsahuje identifikátor id uživatele, jeho e-mail a hashované heslo. Uživatel může mít přiřazené klienty prostřednictvím relační tabulky. Klient obsahuje identifikátor id klienta, jeho název, IČ a DIČ. Klient je přiřazen uživatelům přes vazební tabulku, která obsahuje id klienta a id uživatele. Sklad obsahuje identifikátor id skladu, jeho název, popis a id klienta, ke kterému patří. Každý sklad

může mít více pozic. Pozice obsahuje identifikátor id pozice, její kód a id skladu, ve kterém se nachází. Pozice slouží pro umístění krabic. Produkt obsahuje identifikátor id produktu, jeho název, SKU kód, technické parametry a id klienta, ke kterému náleží. Produkt může mít přiřazeno více šarží. Šarže obsahuje identifikátor id šarže, její název, datum expirace a id produktu, ke kterému patří. Každá šarže může být využita v operaci. Krabice (balení) obsahuje identifikátor id krabice, EAN, šířku, výšku, hloubku, váhu a id pozice, na které se nachází. Každá krabice může být součástí operace. Skladová operace obsahuje identifikátor id operace, typ operace (např. příjem, výdej), datum vytvoření a id uživatele, který operaci vytvořil. Každá operace obsahuje jednu nebo více položek. Položka operace (skupina) obsahuje identifikátor id, id operace, id produktu, množství, a volitelně id šarže a id krabice, do které má být položka vložena. Historie obsahuje identifikátor id záznamu, čas provedení, typ události, id uživatele, který akci provedl, a id související entity, ke které se akce vztahuje (například produkt, krabice, pozice nebo operace).

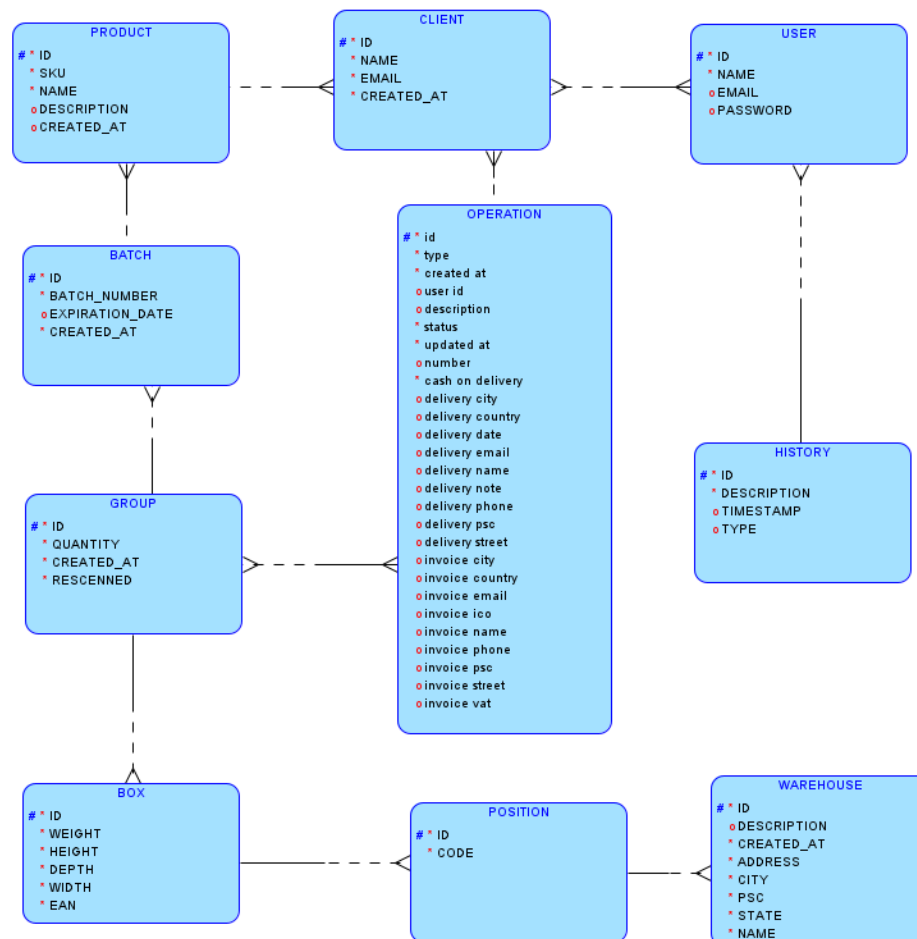
#### **3.4.1.5. Integritní omezení**

- IO 01. Každý záznam v tabulce product\_product musí obsahovat platný odkaz na klienta (client).
- IO 02. Každý záznam v tabulce position\_position musí obsahovat platný odkaz na sklad (warehouse).
- IO 03. Každý záznam v tabulce batch\_batch musí obsahovat platný odkaz na produkt (product).
- IO 04. Každý záznam v tabulce group\_group musí obsahovat platný odkaz na šarži (batch), a může mít odkaz na krabici (box).
- IO 05. Každý záznam v tabulce operation\_operation musí obsahovat platný odkaz na klienta (client).
- IO 06. Každá operace (operation) může mít přiřazeny skupiny (group) přes M:N vztah.
- IO 07. Každý záznam v tabulce history\_history musí obsahovat type, related\_id a může obsahovat odkaz na uživatele (user).
- IO 08. Každý záznam v tabulce user\_user\_client (vazební tabulka) musí obsahovat platné odkazy na uživatele (user) a klienta (client).
- IO 09. Uživatel musí mít unikátní e-mail.
- IO 10. Každý produkt musí mít unikátní SKU (sku).
- IO 11. Pole status v tabulce operation\_operation musí odpovídat jedné z definovaných hodnot (CREATED, BOX, COMPLETED, CANCELLED).

#### **3.4.2. ERD**

Skladový systém obsahuje klienta, který může mít více operací. Klient může mít více operací, ale jedna operace může náležet pouze jednomu klientovi. Uživatel je osoba, která může vytvářet operace. Jedna operace může být vytvořena pouze jedním uživatelem, ale uživatel může vytvořit více operací. Operace obsahuje informace o doručení a fakturaci a zahrnuje více skupin (group). Každá skupina spadá pod více operací a jedna operace může obsahovat více skupin. Skupina reprezentuje konkrétní

množství produktů dané šarže v konkrétní krabici. Skupina se váže na produktovou šarži (batch), která je odvozena z konkrétního produktu. Každá šarže náleží pouze jednomu produktu, ale jeden produkt může mít více šarží. Skupina zároveň odkazuje na jednu krabici, ale jedna krabice může být použita ve více skupinách. Produkt má základní identifikaci pomocí SKU a názvu. Produkt může mít více šarží, ale každá šarže může náležet pouze jednomu produktu. Krabice (box) má parametry jako hmotnost a rozměry. Každá krabice může obsahovat více skupin, ale skupina může být pouze v jedné krabici. Každá krabice má také pozici ve skladu, která je určena přes tabulku position. Pozice má svůj kód a může být sdílena více krabicemi, ale každá krabice může být pouze na jedné pozici. Sklad (warehouse) obsahuje pozice. Každá pozice náleží právě jednomu skladu, ale sklad může obsahovat více pozic. Historie zachycuje události systému, přičemž jedna událost může být zapsána v historii jako záznam. Každý záznam v historii se vztahuje k nějakému typu operace nebo události, ale typ může být sdílen více záznamy. Uživatel provádí akci uloženou do historie. Uživatel může vytvořit více záznamů do historie, ale záznam v historii může být vytvořen pouze jedním uživatelem.

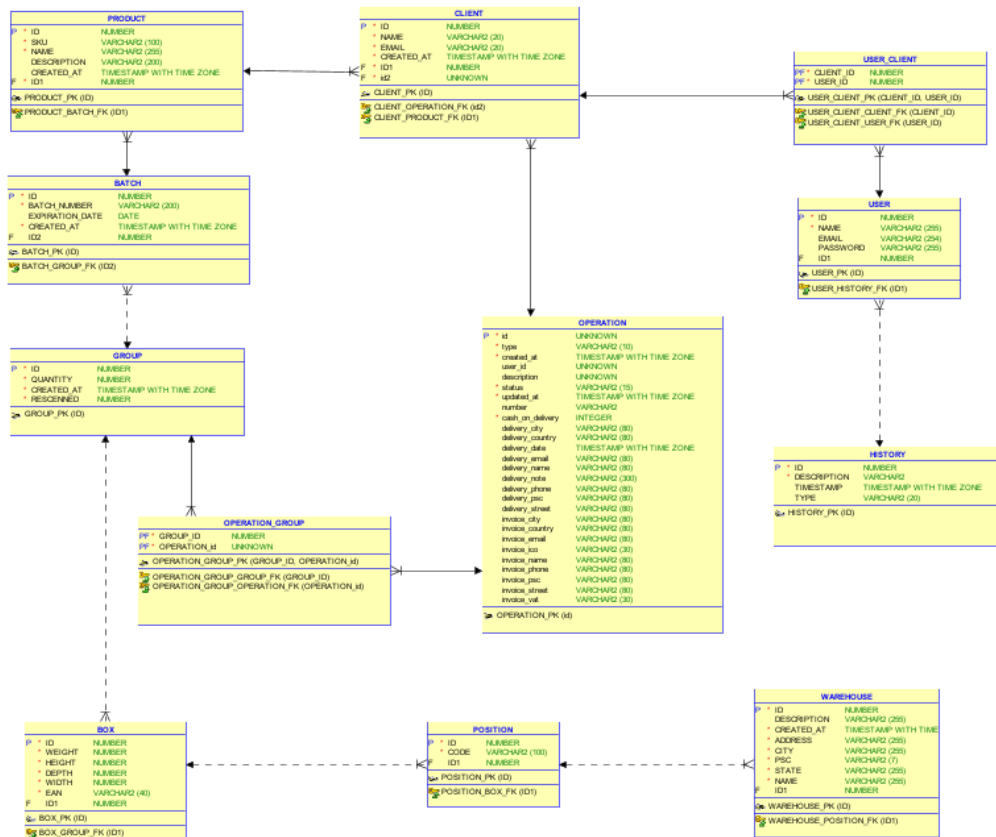


Obrázek 5 ERD diagram Zdroj: vlastní

### 3.4.3. Relační model dat

Relační model odpovídá logickému modelu. V rámci transformace do relační podoby došlo k několika úpravám a zavedení pomocných tabulek. Vazby typu N:M byly rozloženy pomocí přechodových tabulek. Mezi entitami USER a CLIENT vznikla přechodová tabulka USER\_CLIENT, která umožňuje, aby jeden uživatel mohl pracovat s více klienty a jeden klient mohl být přiřazen více uživatelům. Mezi entitami OPERATION a GROUP vznikla přechodová tabulka OPERATION\_GROUP, která propojuje operace s jednotlivými skupinami produktů. Vztahy mezi entitami byly zachovány pomocí cizích klíčů, např. PRODUCT\_ID v tabulce BATCH, BATCH\_ID v tabulce GROUP nebo BOX\_ID v tabulce POSITION. V tabulce OPERATION je zahrnut cizí klíč USER\_ID, který odkazuje na uživatele, který danou operaci provedl, a také CLIENT\_ID, který ji váže ke konkrétnímu klientovi. Každá tabulka má definovaný primární klíč (např. ID), který jednoznačně identifikuje záznam.

Závěrem lze říct, že relační model přesně reflektuje strukturu logického modelu, přičemž byly přidány přechodové tabulky pro vztahy typu N:M a zajištěna referenční integrita pomocí cizích klíčů.



Obrázek 6 Relační model Zdroj: vlastní

### 3.5. Architektura systému

Návrh a realizace systému vychází z architektury klient-server s důrazem na oddělení prezentační a aplikační vrstvy. Celé řešení je rozděleno do tří hlavních částí: frontendové aplikace, backendové aplikace a databázového uložště.

Systém je koncipován jako monolitická aplikace s jasným oddělením odpovědnosti mezi klientskou a serverovou část. S ohledem na zvolené technologie a způsob nasazení není systém rozdělen na mikroservisy, ale tvoří jeden ucelený celek, jehož komponenty spolu komunikují prostřednictvím REST API.

Frontend je implementován pomocí knihovny React ve frameworku Next.js. Ten poskytuje rozšířenou funkcionalitu pro server-side rendering a routing. Pro vývoj uživatelského rozhraní je využita komponentová knihovna MUI (Material UI), která zajišťuje konzistentní a moderní vzhled.

Backend je realizován pomocí webového frameworku Django v jazyce Python. Obsahuje veškerou aplikační logiku, správu dat, autentizaci, práci s AI službami a rozhraní pro komunikaci s databází a frontendem.

Databáze je umístěna na platformě Supabase, která poskytuje PostgreSQL databázi v cloudu. Backend se k ní připojuje přímo pomocí ORM nástroje Django a zajišťuje perzistenci dat.

#### **Komunikace mezi frontendem a backendem:**

1. Klientská aplikace odesílá HTTP požadavky (např. GET, POST, PUT, DELETE) na definované endpointy backendu.
2. Backend tyto požadavky přijímá pomocí funkčních (function-based views) nebo třídních (class-based views) pohledů, které jsou registrovány v souboru `urls.py` webového frameworku Django.
3. Každý požadavek je zpracován konkrétní view funkcí – ta může provádět validaci, volání aplikační logiky, interakci s databází, a nakonec vrátí odpověď ve formátu JSON.

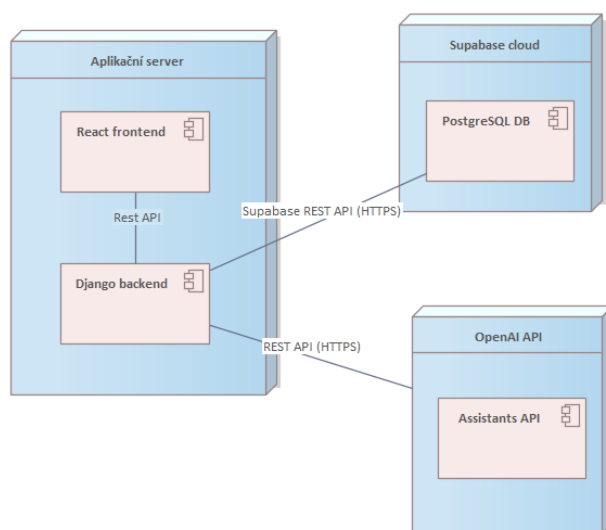
#### **Komunikace mezi backendem a databází:**

1. ORM (Object-Relational Mapping) zajišťuje abstraktní vrstvu pro práci s databázovými tabulkami jako s Python objekty, čímž zjednodušuje databázové operace.
2. Databázové operace jsou prováděny automaticky podle definovaných modelů v Django, které odpovídají tabulkám v databázi.

#### **Komunikace mezi backendem a OpenAI:**

1. Backend odešle uživatelský prompt specifickému asistentovi.

2. Asistent (OpenAI) prompt zpracuje a vytvoří odpověď, která může obsahovat požadavek na volání externí metody (např. funkce implementované v backendu).
3. Backend zachytí požadavek na konkrétní metodu, zpracuje potřebná data a odešle zpět odpověď obsahující výsledek volané metody.
4. Asistent odpověď backendu opět zpracuje, integruje získaná data a vytvoří finální výstup.
5. Backend doručí finální odpověď uživateli.



Obrázek 7 Deployment diagram Zdroj: vlastní

### 3.6. Backendová část systému

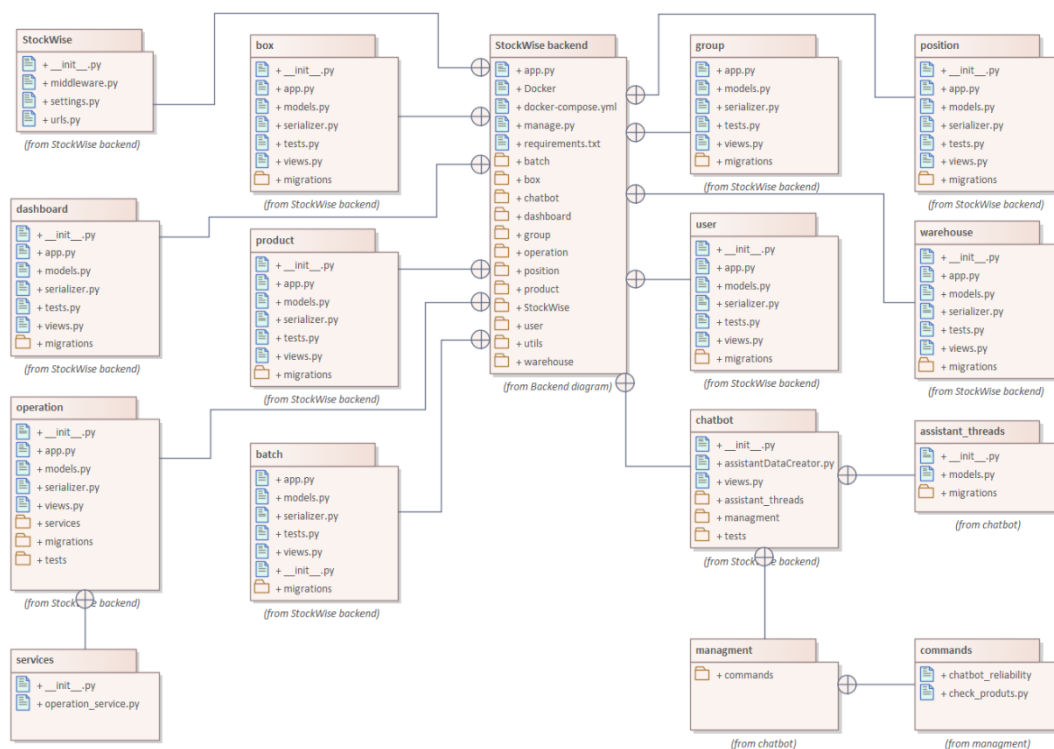
Backendová část systému je implementována ve frameworku Django, který je napsán v jazyce Python a poskytuje robustní prostředí pro vývoj webových aplikací a REST API. Ačkoli Django umožňuje vývoj plnohodnotných webových aplikací za použití vlastního šablonovacího systému (Django Templates), tento přístup nebyl v práci využit.

Důvodem je oddělení klientské a serverové části systému, což je v moderním vývoji běžnou praxí. Django Templates jsou primárně zaměřeny na práci s HTML a CSS a neumožňují pokročilejší interaktivní logiku na úrovni klienta. Stylování a dynamické chování stránky je v tomto přístupu často složité a neflexibilní.

Tento přístup přináší řadu výhod jako jsou oddělení vývoje, snadnější testování a rozšiřování API, možnost využití stejného backendu pro více různých frontendových klientů, jako je web a mobilní aplikace, a využití moderních nástrojů a knihoven na straně klienta (MUI, React Hooks, server-side rendering s Next.js).

### 3.6.1. Struktura

Projekt je rozdělen do logicky oddělených Django aplikací dle jednotlivých doménových oblastí. Každá aplikace obsahující soubor *models.py* a složku migrací reprezentuje tabulku v databázi. Aplikace obsahuje standartní soubory jako je *models.py*, *views.py*, *serializers.py*, *urls.py* a případně složky jako *services/* či *tests/*, dle potřeby.



Obrázek 8 Backend struktura aplikace StockWise Zdroj: vlastní

### 3.6.2. REST API

Komunikace mezi frontendem a backendem probíhá výhradně pomocí REST API postaveného na knihovně Django REST Framework. API je implementováno pomocí viewsetů, přičemž každý viewset reprezentuje kolekci endpointů vztahujících se k jedné konkrétní entitě. Například *OperationViewSet* spravuje všechny operace spojené s objednávkami, jako je vytváření, aktualizace, mazání i vyhledávání, prostřednictvím akcí definovaných pomocí dekorátoru *@action*.

Každý endpoint vrací odpověď ve formátu JSON a vyžaduje autentizaci prostřednictvím JWT tokenů uložených v HTTP-only cookies, čímž je minimalizováno riziko zneužití na straně klienta. Celé API je navrženo tak, aby bylo bezpečné, snadno rozšiřitelné a jednoduše integrovatelné s frontendem i dalšími systémy. Struktura endpointů respektuje principy REST architektury, přičemž specifické operace (např. vyhledávání, změna stavu operace, práce s krabicemi) jsou řešeny pomocí vlastních akcí v rámci příslušných viewsetů.

### 3.6.2.1. Autentizace

Autentizační vrstva systému je postavena na JWT a Django REST Framework. Poskytuje bezpečné, robustní a uživatelsky přívětivé řešení správy přihlášení, registrace a správy hesel.n

Komunikace s frontendem je reslitována pomocí samostatných REST API endpointů. Všechny JWT tokeny (access a refresh) jsou ukládány jako http-only cookies.

Endpoint `/api/user/register/` – Registrace uživatele

- Metoda: POST
- Popis: Vytvoření nového uživatelského účtu na základě e-mail adresy a silného hesla.
- Bezpečnostní opatření: Povinná validace síly hesla (minimální délka, číslo, velké písmeno, speciální znak)
- Statusy: 201 Created, 400 Bad Request

Endpoint `/api/user/login/` – Přihlášení uživatele

- Metoda: POST
- Popis: Ověření uživatelských údajů a vydání access a refresh tokenu. Tokeny jsou vráceny v bezpečnostních cookies (HttpOnly, Secure, SameSite=None)
- Statusy: 200 OK, 401 Unauthorized

Endpoint `/api/user/logout` – Odhlášení uživatele

- Metoda: POST
- Popis: Odstranění cookies s access a refresh tokeny, ukončení sezení
- Statusy: 200 OK

Endpoint `/api/user/refresh_token/` – Obnovení access tokenu

- Metoda: POST
- Popis: Vydání nového access tokenu na základě platného refresh tokenu uloženého v cookies.
- Statusy: 200 OK, 401 Unauthorized

Endpoint `/api/user/me /` – Získání přihlášeného uživatele

- Metoda: GET
- Popis: Vrátí informace o aktuálně přihlášeném uživateli
- Statusy: 200 OK

Endpoint `/api/user/ request_password_reset /` – Žádost o reset hesla

- Metoda: POST
- Popis: Odeslání odkazu pro reset hesla na registrovaný email uživatele. Reset link obsahuje krátkodobý JWT token typu `password_reset`, platný 15 minut.
- Statusy: 200 OK, 404 Not Found

Endpoint `/api/user/ reset_password /` – Reset hesla

- Metoda: POST
- Popis: Uživatel nastaví nové heslo pomocí odkazu získaného e-mailem. Token ověřuje platnost a typ požadavku
- Statusy: 200 OK, 400 Bad Request

Technické detaily

- Validace síly hesla je řešena pomocí vlastní funkce `is_strong_password()`, která ověřuje požadovanou složitost hesla.
- Registrace a přihlášení využívají Django model `User`, který je serializován pomocí `UserSerializer`.
- Správa tokenů je realizována pomocí knihovny `django-rest-framework-simplejwt`.
- Reset hesla je bezpečně řešen pomocí krátkodobých JWT tokenů, které nesou specifický typ (`password_reset`)
- Bezpečnostní opatření:
  - Využití `HttpOnly`, `Secure`, a `SameSite=None` cookies pro všechny tokeny
  - Časově omezené tokeny pro reset hesla (15 minut)

### 3.6.2.2. Serializery

Každý model systému využívá vlastní serializer, který odpovídá specifickým požadavkům, ať už se jedná o běžný zápis dat, rozšířené zobrazení s agregovanými informacemi, nebo hromadné operace.

Serializery slouží jako prostředník mezi modelem a výsledným JSON výstupem, který je předáván frontendové aplikaci. Umožňují nejen mapování polí modelu, ale i jejich transformaci, obohacení o dynamicky generované hodnoty.

Přístupy ke všem relačním polím jsou optimalizovány pomocí efektivního `select_related` a `prefetch_related`.

Ukázkový příklad – Produkt

Pro entitu `Produkt` jsou definovány dva hlavní serializery a to `ProductSerializer` a `ProduktBulkSerializer`.

## ProduktSerializer

Používá se pro detailní zobrazení produktu na frontendu, včetně agregovaných informací o navázaných šaržích (*batch*) a skupinách (*group*).

Tabulka 2 Atributy serializeru *ProductSerializer* Zdroj: vlastní

Atribut	Popis
id	ID produktu
client_id	ID klienta, vazba přes PrimaryKeyRelatedField na entitu Client
batches	Dynamicky generované pole obsahující počet šarží a jejich identifikátory
groups	Dynamicky generované pole obsahující počet skupin a jejich identifikátory
amount_cached	Předepsané množství produktu dostupné ve skladu
created_at	Datum vytvoření produktu
name	Název produktu
sku	Jedinečný kód produktu
description	Popis produktu

Metody *get\_batches* a *get\_groups* jsou implementovány pomocí *SerializerMethodFiel*s, které umožňují komplexní sestavení strukturovaných dat.

Pole *amount\_cached*, *batches* a *groups* jsou definovaná pouze pro čtení.

## ProductBulkSerializer

Minimalistická varianta serializeru určená pro hromadné vytvoření více produktů najednou, například při importu dat.

Tabulka 3 Atributy serializeru *ProductBulkSerializer* Zdroj: vlastní

Atribut	Popis
client_id	ID klienta
name	Název produktu
sku	Unikátní kód produktu
description	Popis produktu

Obsahuje pouze klíčová pole potřebná pro základní definici produktu. Metoda *create* provádí efektivní hromadné vložení (*bulk\_create*).

Použití samostatných serializerů pro různé účely přispívá k lepší srozumitelnosti API, optimalizaci výkonu a jednodušší správě dat na straně backendu. Serializery zároveň umožňují sestavovat komplexní struktury dat tak, aby přesně odpovídaly potřebám frontendové aplikace, a to bez nadbytečných dotazů na databázi.

### 3.6.2.3. Stránkování

Systém využívá vlastní stránkovací mechanismus definovaný pomocí třídy *CustomPageNumbrPagingation*, která rozšiřuje standartní stránkování Django REST Framework.

Stránkování umožňuje efektivní práci s velkými objemy dat a zároveň poskytuje možnost přepnout mezi stránkovaným a nestránkovaným režimem.

Ve výchozím nastavení je odpověď rozdělena do stránek:

- Výchozí velikost stránky je 10
- Maximální velikost stránky je omezena na 100
- Velikost stránky lze ovlivnit parametrem *page\_size* v URL dotazu

Systém umožňuje na základě parametru *no\_page=1* úplně vypnout stránkování a vrátit všechny položky v jedné odpovědi. Tento režim je vhodný například při exportu dat nebo při malých objemech dat, jako je například získání seznamů klientů uživatele, kdy je nežádoucí rozdělovat odpověď na menší stránky.

```
{
  "count": 135,
  "results": [
    {...}, {...}, {...}
  ]
}
```

Zdrojový kód 5 Struktura stránkované odpovědi Zdroj: vlastní

Na rozdíl od standartního stránkování nejsou vrácena pole *next* ani *previous*, což zjednodušuje strukturu odpovědi.

### 3.6.2.4. Operace

Operace představují středobod celé aplikace a tvoří jednu z nejdůležitějších logických vrstev systému. Slouží k řízení procesů jako je příjem, výdej, balení a přesun produktů mezi krabicemi.

Jejich návrh je postaven na kombinaci Django REST framework, dedikovaných service funkcí a pečlivě oddělené business logiky.

Všechny databázové změny v rámci vytvoření operace probíhají v rámci atomické transakce, aby bylo zajištěno, že v případě chyby nedojde k častému zápisu. Veškeré chyby, jako jsou nedostatek zásob a duplicita šarže, jsou ošetřeny pomocí výjimek a detailních chybových hlášek.

## Architektura operací

### ViewSet – OperationViewSet

Základ komunikace zajišťuje *OperationViewSet*, který vychází z *ReadOnlyModeViewSet*. Umožňuje přehledové zobrazení (*list*), detailní náhled (*retrieve*) a přidává vlastní akce pro vytváření (*create*), vyhledávání (*search*), aktualizaci (*update*), mazání (*remove*), správu stavu a operace s produkty a krabicemi (*add\_to\_box*, *close\_box*, *start\_packiging*, *complete\_packiging*).

### Servisní vrstva – operation\_service.py

Veškerá bussiness logika operací je rozdělena na servisní funkce, čímž dochází k čistému oddělení mezi prezentační a logickou vrstvou.

### Klíčové funkce:

- *create\_operation()* - vytváření operací typu *IN* nebo *OUT*
- *add\_group\_to\_in\_operation()* – přidání nového produktu do příjemky, včetně šarže a skupiny
- *add\_group\_to\_out\_operation()* – přidá produkt do výdejky s podporou rozdělování množství mezi více skupin
- *add\_product\_to\_box()* - přidává produkty do krabice a správně rozdělí množství
- *get\_operation\_product\_summary()* – vrací agregovaný přehled produktů v rámci operace
- *update\_operation()* – aktualizuje parametry operace na základě vstupních dat
- *remove\_operation()* – bezpečně odstraní operaci a s ní spojené entity (v závislosti na typu operace)

### Proces vytváření operace:

1. Validace vstupu: Ověření typu operace, existence klienta, správnost produktů
2. Založení operace: Vytvoření objektu *Operation* se stavem *CREATED*
3. Zpracování produktů
  - a. Příjemka (*IN*): Vytvoření nové šarže, skupiny a volitelně i nové krabice
  - b. Výdejka (*OUT*): Vyhledání existujících zásob, rozdělení množství, propojení se skupinami.
4. Uložení doručovacích/fakturačních údajů v případě výdejky
5. Vracení ID nově vytvořené operace

## 3.7. Implementace chatbota

### 3.7.1. Logika chatbota

Základem interakce uživatele se systémem pomocí přirozeného jazyka je třída *OpenAIHandler*, která zapouzdřuje komunikaci s rozhraním OpenAI API. Tato třída obsahuje odesílání promptů, správu vláken (*thread\_id*) a běhu (*run\_id*) včetně zpracování odpovědí.

Metoda *get\_or\_create\_threads* a *reset\_threads*

Každý uživatel má své vlastní konverzační vlákno, které je identifikováno kombinací uživatele, klienta a modelu. Pokud vlákno neexistuje, je vytvořeno a uloženo do databáze. Resetování konverzací probíhá tak, že se vytvoří nové *thread\_id* a staré vlákno se přepíše.

Metoda *cancel\_active\_runs*

Funkce *cancel\_active\_runs* zruší všechny rozběhlé runy, které ještě nejsou dokončené, protože by mohly kolidovat s novým požadavkem. Na začátku jsou načteny aktivní běhy (*queued*, *in\_progress*, *requires\_action*) a ty jsou následně zrušeny pomocí *cancel* metody. Opakovaně se ověřuje stav až do úspěšného zrušení nebo timeoutu. Tento proces je nezbytný pro stabilitu, protože OpenAI nepodporuje souběžné běhy ve stejném vlákně.

Metody *wait\_for\_completion* a *handle\_tool\_calls*

Metoda *wait\_for\_completion* umožňuje čekání, dokud aktuální běh nedoběhne nebo selže, případně nevyžádá akci. Sleduje stav běhu přes *run\_id* a pokud je vyžadována akce „*requires\_action*“, zavolá se *handle\_tool\_calls*, který zpracuje požadavky na volání nástrojů (function calling). Logika zpracování vlastních funkcí bude popsána v následující kapitole.

Metoda *get\_response*

Metoda vrací odpověď po dokončení běhu. Textové odpovědi vrací jako HTML div, zatímco obrázkové či grafové odpovědi stáhne a převede na base64, aby je bylo možné zobrazit na frontendu.

Metoda *get\_thread\_messages*

Metoda *get\_thread\_messages* je využívána pro získání historie konverzace daného vlákna. Využívá se především při vykreslení komponentu.

### 3.7.1.1. Hlavní metoda *run\_prompt*

Zde je prováděna hlavní logika společná pro statistiky a chatbota, která spouští metody popsané v předešlých kapitolách. Nejprve zkontroluje, zda pro daný dotaz/statistiku neběží jiný běh, poté načte nebo vytvoří vlákno. V případě potřeby zruší aktivní běhy. Následně pošle prompt, spustí běh a počká na výsledek. Nakonec vrátí odpovídající odpověď.

### 3.7.1.2. API rozhraní pro frontend

Logika interakce mezi uživatele a umělou inteligencí je v aplikaci soustředěna do dvou hlavních tříd, kterými jsou *ChatbotView* a *StatisticsView*. Obě třídy jsou odvozené z API View poskytované Django REST Frameworkem a slouží jako vstupní bod pro požadavky z frontendu, ať už jde o běžnou konverzaci (*ChatbotView*) nebo generování specifických statistických výstupů (*StatisticsView*)

Třída *ChatbotView* zajišťuje zpracování všech vstupů, které se týkají obecné konverzace mezi uživatelem a chatbotem. Ve své metodě *post* nejprve načítá základní parametry ze vstupního požadavku jako jsou identifikátor klienta, samostatný textový vstup (prompt) a případně nahraný soubor ve formátu .csv nebo .xlsx. Dále obsahuje i pomocné příkazy jako reset konverzace (*reset=true*) nebo požadavek na zobrazení historie (*history=true*).

V případě resetu je pomocí instance třídy *OpenAIHandler* zavolána metoda *reset\_thread*, která zajistí vytvoření nové konverzace. Pokud je požadována historie, je využita metoda *get\_thread\_messages*.

Třída *StatisticsView* pracuje na velmi podobném principu, ale zaměřuje se výhradně na zpracování specifických typů statistických dotazů. Namísto volného vstupu uživatele se zde očekává předem definovaný *stat\_id*, který slouží jako klíč pro výběr odpovídajícího promptu ze slovníku *STAT\_PROMPTS*. Každý prompt je navržen tak, aby výsledkem byla konkrétní forma výstupu, ať už tabulka nebo graf.

```
"stockSummary": (  
    "Vygeneruj aktuální přehled stavu zásob na skladě. "  
    "Rozděl zásoby podle produktů a jejich šarží. "  
    "U každého záznamu uveď název produktu, označení šarže, aktuální  
množství a jednotku. "  
    "Zobraz výstup **pouze jako tabulku**. Nepoužívej grafy."  
),  
  
"expiringSoon": (  
    "Zobraz produkty, kterým končí expirace během následujících 30 dnů. "  
    "Uveď název produktu, šarží a datum expirace. "
```

"Zobraz **\*\*pouze tabulku\*\***. Nepoužívej žádný text ani graf."

),

Zdrojový kód 6 Ukázka statistického promptu Zdroj: vlastní

### 3.7.1.3. Komunikace s datovým modelem

Pro komunikaci s datovým modelem aplikace pomocí umělé inteligence slouží třída *AssistantDataCreator*, která zajišťuje vykonání konkrétní akce (získání, vytvoření, úprava nebo hromadné vytváření dat) na základě dotazu zpracovaného chatbotem. Tato třída je úzce provázána s mechanismem OpenAI function calling, tedy voláním nástrojů definovaných na backendu, které dokáže OpenAI spustit v reakci na uživatelský prompt. Tímto způsobem lze z prostředí přirozeného jazyka přistupovat ke konkrétním datům v systému a provádět nad nimi operace.

#### Metoda *get\_data*

Tato metoda zajišťuje načtení dat z databáze pro libovolný model. Na základě vstupních parametrů (code, ean, datумы...) sestaví dotaz a provede filtrování. Pokud je přítomen parametr *onlyCount*, vrací se pouze počet odpovídajících záznamů. Jinak jsou data serializována a vrácena jako odpověď. Metoda podporuje i základní filtrování podle klienta.

#### Metoda *create\_data*

Metoda *create\_data* slouží k vytvoření jednoho záznamu na základě parametrů předaných chatbotem. Do vstupních dat automaticky doplňuje identifikaci uživatele a klienta. Po validaci pomocí příslušného serializeru dochází k vytvoření objektu a vrácení zprávy o úspěchu nebo chybě.

#### Metoda *bul\_create\_data*

Pro hromadné vkládání dat slouží metoda *bulk\_create\_data*. Očekává seznam objektů ve vstupním poli *items*. Každému záznamu doplní informace o uživateli a klientovi. Pomocí serializeru se data validují a poté se pomocí metody *bulk\_create* efektivně uloží do databáze. Výstupem je počet vytvořených objektů nebo validační chyby.

#### Metoda *update\_data*

Metoda *update\_data* slouží k úpravě existujícího záznamu. Na základě zadaného id se pokusí vyhledat objekt a poté nad ním provede částečnou aktualizaci. Využívá k tomu serializer s *partial=True*. Pokud záznam neexistuje, je vrácena chyba.

### Metoda *get\_function*

Tato funkce mapuje názvy nástrojů, které jsou volány umělou inteligencí (např. *createProduct*, *getUsers*, *updateOperation*) na konkrétní metody třídy *AssistantDataCreator*, včetně odpovídajícího modelu a serializery. Výstupem je trojice, kterou *OpenAIHandler* použije ke zpracování požadavku.

### Další pomocné metody

Třída *AssistantDataCreator* obsahuje také několik doplňkových metod, které rozšiřují možnosti interakce s daty. Metoda *get\_history\_data* načítá historii změn záznamů s podporou filtrování podle typu a časového rozsahu. Na ni navazuje *get\_specific\_history\_data*, která předem definuje typ (např. operace, produkt) a zjednodušuje tak dotazování.

Pro časové filtrování jsou připraveny metody *filter\_by\_year*, *filter\_by\_month* a *filter\_by\_day*, které umožňují omezit výsledky dle roku, měsíce nebo dne vytvoření záznamu. Nakonec metoda *get\_operation\_statuses* vrací výčet možných stavů operací definovaných v modelu, což se hodí např. pro validaci nebo filtrování v chatbotu.

## 3.8. Frontendová část systému

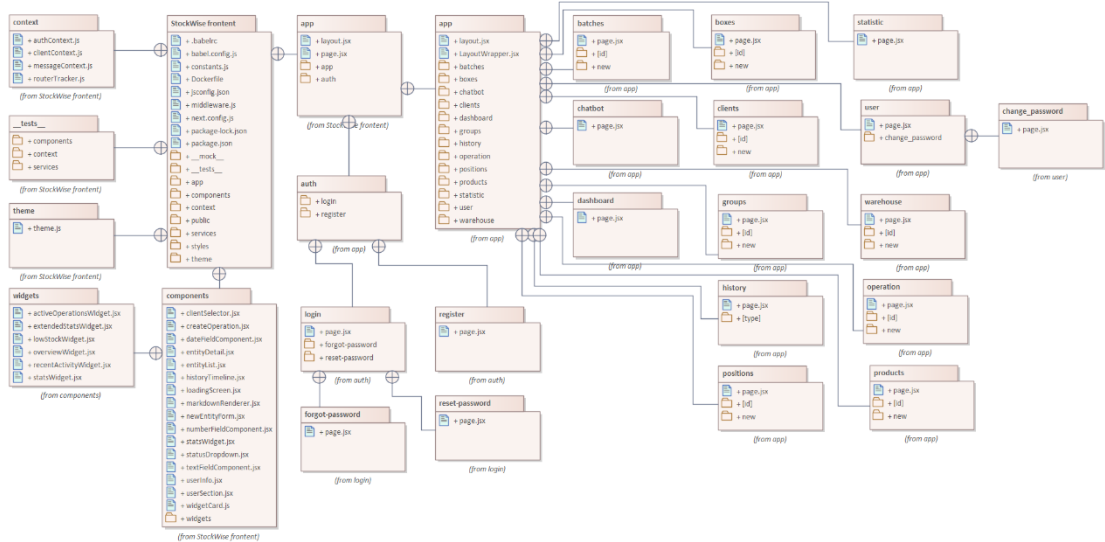
Jak již bylo zmíněno, frontendová část aplikace je implementována pomocí knihovny React ve spojení s frameworkem Next.js. Využívá komponentový přístup a přehlednou architekturu složenou ze složek jako *components*, *context*, *services*, *styles* a dalších.

### 3.8.1. Struktura aplikace

Struktura aplikace je navržena modulárně tak, aby byla přehledná, škálovatelná a dobře testovatelná.

Nejdůležitější složky:

- *components/* – obsahuje znovupoužitelné komponenty jako *EntityList*, *EntityDetail*, *TextFieldComponent* apod.
- *context/* – globální kontexty jako *useAuth*, *useClient*, *useMessage*, které zajišťují sdílení stavů napříč aplikací.
- *services/* – služby pro komunikaci s backendem, postavené nad axios klientem (*apiClient*).
- *app/* – hlavní logika Next.js aplikace.



Obrázek 9 Frontend struktura aplikace StockWise Zdroj: vlastní

## Služby (services)

Frontendové služby slouží jako mezivrstva mezi uživatelským rozhraním a backendovým API. Zajišťují odesílání požadavků, příjem odpovědí a centralizují logiku komunikace s datovým rozhraním serveru, čímž výrazně zjednodušují práci komponent a zvyšují udržitelnost kódu.

Základem služeb je funkce *createCRUDService*, která poskytuje standardní CRUD metody pro libovolný endpoint. Tato služba může být následně rozšířena o jiné metody.

## Komponenty (components)

Jedním z hlavních principů aplikace je znovupoužitelnost komponent. Mezi nejdůležitější a nejuniverzálnější komponenty patří *EntityList*, která slouží k výpisu libovolných entity v podobě tabulky. Má zabudované stránkování, server-side vyhledávání, možnost filtrování a akce jako je zobrazení detailu či mazání záznamu.

Druhou základní komponentou je *EntityDetail*, která zobrazuje detail entity s možností úpravy. Dynamicky generuje formulářová pole podle typu (text, number, date, select). Využívá vlastní komponenty *TextFieldComponent*, *NumberFieldComponent* a *DateFieldComponent*.

## Kontext (context)

V aplikaci jsou pro správu globálního stavu využívány React Contexty, které umožňují efektivní sdílení dat mezi různými komponentami bez nutnosti prop-drillingu. Každý kontext má jasně definovanou odpovědnost a zajišťuje logiku související s určitou oblastí aplikace. Jsou využívány pro uchování klientů, přihlášeného uživatele a chybových hlášek.

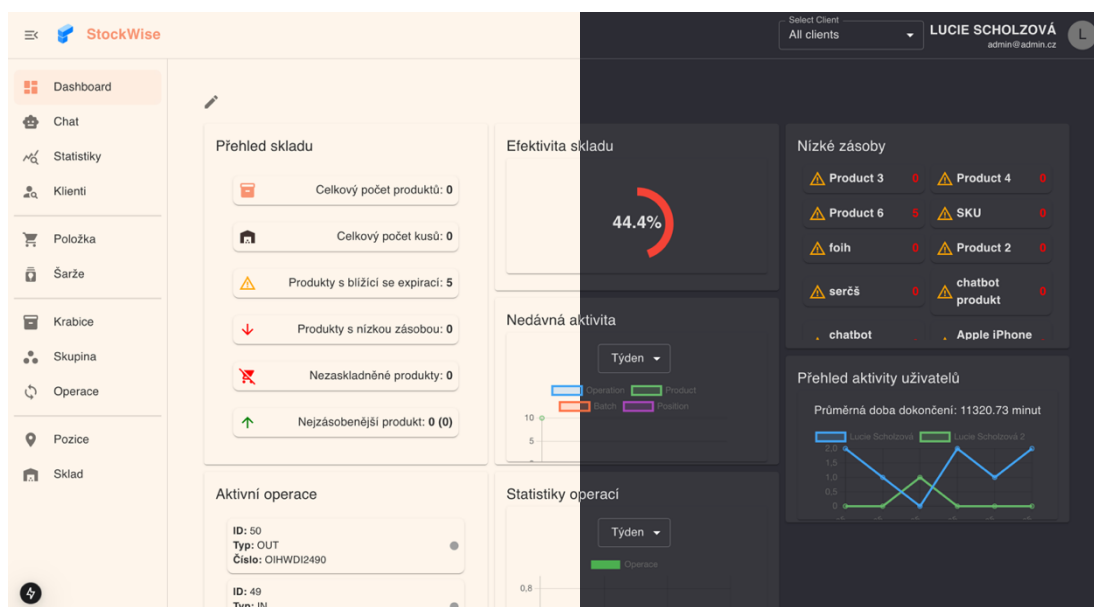
useClient

Kontext *ClientContext* slouží k uchování a správě aktuálně vybraného klienta. Tato funkcionality se využívá především k filtrování zobrazených dat (například produktů) podle daného klienta. Stav se uchovává v *sessionStorage*, aby zůstal zachován i po obnovení stránky. Hodnoty *selectedClient* a *setSelectedClient* jsou dostupné ve všech komponentách zabalených v *ClientProvider*.

### 3.8.2. Uživatelské rozhraní

#### Barvy a layout

Uživatelské rozhraní aplikace StockWise je navrženo s důrazem na přehlednost, estetiku a funkčnost při každodenním používání. Vzhled aplikace je konzistentní, responzivní a podporuje přepínání mezi světlým a tmavým režimem.



Obrázek 10 Tmavý a světlý mód Zdroj: vlastní

Aplikace využívá vlastní definovanou paletu pro světlý i tmavý režim. Světlý režim využívá teplé tóny jako tlumenou oranžovou #fca658 a přírodní béžovou #fff3ed, které působí přívětivě a snižují vizuální únavu. Tmavý režim využívá elegantní odstíny šedé a kontrastní oranžové a žluté, např. #f89371 a #F4D35E, pro zvýraznění prvků a pohodlné používání v nočním prostředí.

Rozhraní je děleno na tři části. Postranní menu slouží k navigaci mezi moduly aplikace (Dashboard, Chat, Operace...) a umožňuje skrytí pro větší přehlednost hlavního panelu. Horní panel obsahuje přepínač klienta, přihlášeného uživatele a přístup k nastavení. Hlavní panel zobrazuje dashboards, formuláře, seznamy a grafy. Díky komponentovému přístupu je snadné tuto část přizpůsobit či rozšířit o nové moduly.

Aplikace je plně responzibilní a optimalizovaná pro různé velikosti obrazovek a umožňuje uživateli přepínat mezi dark a light módem. Přepínač je dostupný v sekci uživatele.

### Dashboard

Úvodní stránka aplikace je tvořena přehledným dashboardem, který slouží jako hlavní rozcestník a zároveň poskytuje uživatele okamžitý přehled o klíčových ukazatelích skladu. Dashboard je sestaven z modulárních widgetů, které lze snadno přizpůsobit potřebám uživatele.

Seznam dostupných widgetů:

- **Přehled skladu:** Poskytuje souhrnné informace o stavu skladu jako je celkový počet produktů a kusů, počet produktů s blížící se expirací, produkty s nízkou zásobou, nezaskladněné produkty a nejzásobenější produkty.
- **Aktivní operace:** Zobrazuje seznam právě probíhajících skladových operací.
- **Efektivita skladu:** grafický a procentuální indikátor, zobrazující poměr dokončených a nedokončených operací.
- **Přehled aktivity uživatelů:** Zobrazuje aktivitu jednotlivých uživatelů.
- **Nízké zásoby:** Tabulka produktů, které se blíží ke kriticky nízkému stavu zásob. Umožňuje filtrování dle časového intervalu.
- **Nedávná aktivita:** Vizualizace posledních změn ve skladu.
- **Statistiky operací:** Sloupcový graf zobrazující přehled operací podle typu a času. Umožňuje filtrování dle časového intervalu.

### Uživatelská stránka

Aplikace obsahuje stránku pro správu uživatelského profilu, která slouží správě přístupových údajů. Stránka je navržena minimalisticky a intuitivně tak, aby uživatelé mohli snadno upravit své osobní údaje nebo přidat či odebrat klienta. Je zde také možnost odhlášení z aplikace.

Dále je zde možnost změnit heslo uživatele Tato funkcionality byla kvůli vyšší bezpečnosti oddělena a v editaci klienta je umístěno pouze tlačítko, které uživatele přesměruje na formulář. Uživatel zde musí zadat staré heslo a následně nové, které pro ověření musí zadat dvakrát.

### Seznam (položek)

Aplikace obsahuje opakovatelně použitelnou šablonu stránky pro zobrazení seznamu různých typů entit, jako jsou produkty, skupiny, krabice, operace a další. Stránka je tvořena především interaktivní tabulkou, která zobrazuje klíčové atributy každé entity. Standardně obsahuje tlačítko pro vytvoření nové entity, které zobrazí stránku pro vytvoření nového záznamu (shodný s editací), dále vyhledávací pole, tlačítka akcí pro mazání/úpravu/historii/balení a referenční hodnoty na jiné navazující entity.

Tabulka umožňuje stránkování pro efektivní zobrazování, kde si uživatel může nastavit kolik záznamů chce na stránku zobrazit.

### Detail produktů

Každá entita v aplikaci má svou vlastní stránku detailu, která slouží k zobrazení kompletních informací o konkrétním záznamu. Cílem této stránky je umožnit rychlý přehled o obsahu dané entity bez nutnosti přecházet do editačního režimu, s možností snadného přechodu na úpravy pomocí tlačítka.

### Historie

Aplikace podporuje plnou auditovatelnost každého záznamu prostřednictvím historie změn. Tato funkce je dostupná pro všechny důležité typy entit, jako jsou operace, produkty, pozice, krabice, šarže a skupiny. Uživatel se na stránku dostane přímo ze seznamu záznamů kliknutím na ikonu historie.

### Balení operace

Stránka balení je klíčovou součástí procesu expedice a slouží k evidenci fyzického přebalení produktu do krabic pro expedici. Je navržena tak, aby umožňovala rychlou, intuitivní a přesnou manipulaci s produkty v reálném čase, ideálně s podporou čtečky čárových kódů.

Stránka umožňuje vybrání aktuální krabice, do které se balí, po vybrání je zobrazen seznam produktů, které v krabici aktuálně jsou. Po vybrání aktuálního produktu se automaticky přednačte zbývající počet daného produktu pro expedici. Toto množství si může skladník změnit podle toho, kolik kusů do krabice ukládá.

Pro zabalení všech produktů v operaci se zobrazí tlačítko na dokončení balení a po kliknutí je aplikace označena jako hotová.

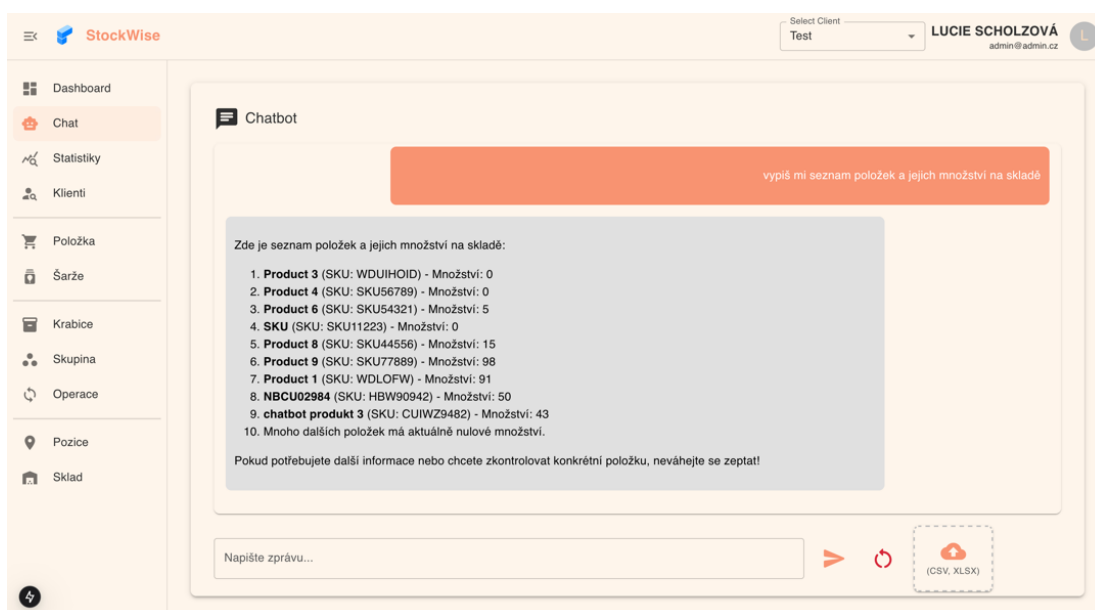
### Chatbot

Jedním z klíčových prvků aplikace je integrovaný chatbot, který využívá umělou inteligenci pro přímou komunikaci s uživatelem. Tento chatbot výrazně zjednodušuje a automatizuje práci ve skladovém systému. Uživatel s ním komunikuje formou běžné přirozené řeči, čímž odpadá složitá formuláře, či nutnost detailní znalosti systému.

Každý uživatel má vlastní vlákno konverzace pro daného klienta, ve které se ukládá historie, díky které se chatbot může danému uživateli přizpůsobit. Aktuální konverzace je umožněna pouze pro aktuálně vybraného klienta, tudíž není možné komunikovat napříč klienty.

Chatbotu umožňuje:

- Zobrazování dat
  - „Vypiš seznam položek a jejich množství na skladě“
- Filtrování podle dotazů v přirozeném jazyce
  - „Kolik máme na skladě produktů, které expirují do týdne?“, „Zobraz produkty s množstvím 0“.
- Real-time komunikaci
- Vytváření nových záznamů
- Mazání a úpravu existující záznamů
  - „Smaž produkt s názvem testovací“, „Změň název produktu SKU1123 na 'Náhradní díl'“.
- Hromadné vytváření produktů pomocí souboru Excel
- Generování grafů a statistik
  - „Vykresli mi sloupcový graf počtu operací za poslední měsíc“, „Jaká je efektivita skladu za poslední týden?“.
- Reset konverzace



Obrázek 11 Stránka chatbota Zdroj: vlastní

## Chatbot statistiky

Chatbot statistiky umožňují dynamické zobrazování připravených analýz skladu ve formě widgetů. Uživatel si může zobrazit základní přehledy i pokročilé predikce podle potřeby. Statistiky jsou generovány na základě aktuálních dat pomocí předdefinovaných promptů.

- Přehled aktuálních zásob produktů a šarží, zobrazených jako tabulka
- Výpis produktů s expirací do 30 dnů, zobrazený tabulkou
- Grafické zobrazení počtu příjmů a výdejů za poslední 3 měsíce, formou čárového grafu se dvěma liniemi

- Vyhodnocení efektivity uživatelů podle počtu provedených operací pomocí sloupcového grafu
- Seznam posledních změn ve skladu za posledních 7 dní, seřazený chronologicky jako seznam nebo tabulka
- Identifikace produktů s nízkým stavem zásob, zobrazená tabulkou
- Přehled produktů s nejvyšším počtem výdejů, zobrazený sloupcovým grafem
- Souhrn klíčových událostí za poslední měsíc (nové produkty, operace, expirace, přesuny) v podobě číselné tabulky
- Predikce očekávané spotřeby produktů na příští měsíc pomocí čárového grafu (max. 10 nejčastějších položek)
- Odhad expirací v následujícím měsíci, zobrazených jako sloupcový graf podle týdnů
- Zhodnocení využití pracovníků, vizualizované pomocí kruhového nebo sloupcového grafu
- Heatmapa zaplněnosti skladu na základě pozic
- Přehled pohybu produktů za poslední dva týdny ve formě čárového nebo sloupcového grafu

Statistiky jsou generovány pomocí umělé inteligence a nelze garantovat, že budou na 100 % přesné. Jejich výsledky doporučuji požadovat za orientační a nezakládat na nich klíčová rozhodnutí. Pravdivost některých údajů lze částečně ověřit porovnáním s některými grafy v hlavním dashboardu. V určitých případech se také může stát, že statistika nebude vůbec vykreslena.

### **3.9. Efektivita využití umělé inteligence**

V rámci praktické části diplomové práce byla testována efektivita umělé inteligence (konkrétně OpenAI Assistant API) pro integraci se skladovým systémem. Výsledky ukazují, že AI může významně usnadnit práci s daty, avšak zároveň přináší i několik zásadních omezení, která je nutné zohlednit při reálném nasazení.

Byly provedeny dva experimenty, kde byl chatbot dotázán 100x na stejný prompt. Experiment byl proveden pomocí python commands, ve kterých byla následně kontrolována správnost odpovědi a výsledek byl uložen do .csv souboru.

#### **3.9.1. Experiment 1 - výpis všech produktů**

V prvním testu byl chatbot požádán, aby vypsal všechny produkty pro klienta s ID 1. Prompt byl následující: “Vypiš seznam produktů pro klienta id 1. Vypiš name produktu. Odpověz jako seznam názvů oddělený čárkami.,,,”

Výsledky ukázaly, že chatbot správně uvedl několik produktů, ale zároveň některé vynechal. Například „Produkt 6“ se v odpovědích vůbec nevyskytl, přestože patří do sady dat pro daného klienta. To může mít závažné důsledky při operacích, které vyžadují úplnost informací.

V následující tabulce jsou první 2 odpovědi.

Tabulka 4 Ukázka odpovědi v prvním experimentu – výpis všech produktů Zdroj: vlastní

run	actual
1	Seznam produktů pro klienta ID 1: ASUS ROG Strix G15, Product 2, Sony WH-1000XM5, WD My Book 8TB, Google Pixel 7, Xiaomi Mi Band 7, LG OLED CX 55", Product 3, Product 4, JBL Flip 6, Product 1, Microsoft Surface Pro 9, NBCU02984, Noctua NH-D15, be quiet! Pure Base 500DX, Samsung 980 Pro 2TB SSD, Fractal Design Meshify 2, Arctic MX-6 Thermal Paste, NZXT Kraken X63, TP-Link RE605X Extender, Baseus GaN Charger 65W, Sony ZV-E10, Logitech G Pro X Headset, Rode NT-USB Mini, LG UltraWide 34WN80C, Apple Magic Keyboard, That, Parent, Partner, Although, Company, Half, Natural, Method, Rich, See, May, Point, Into, Maintain, Yourself, Performance, Suddenly, Final, Heart, Fish, Hundred, Factor, Apple iPhone 14, Samsung Galaxy S22, Lenovo ThinkPad X1, Dell UltraSharp 27", Logitech MX Master 3, Apple MacBook Pro 16", HP LaserJet Pro M404n, Canon EOS R6, TP-Link Archer AX55, Philips Hue Starter Kit, Razer DeathAdder V2, Corsair K95 RGB Platinum, Sony PlayStation 5, Xbox Series X, Apple iPad Air 5, Amazon Kindle Paperwhite, AMD Ryzen 9 7950X, Intel Core i9-13900K, ASUS ZenBook 14 OLED, Synology DS920+.
2	ASUS ROG Strix G15, Product 2, Sony WH-1000XM5, WD My Book 8TB, Google Pixel 7, Xiaomi Mi Band 7, LG OLED CX 55", Product 3, Product 4, JBL Flip 6, Product 1, Microsoft Surface Pro 9, NBCU02984, Noctua NH-D15, be quiet! Pure Base 500DX, Samsung 980 Pro 2TB SSD, Fractal Design Meshify 2, Arctic MX-6 Thermal Paste, NZXT Kraken X63, TP-Link RE605X Extender, Baseus GaN Charger 65W, Sony ZV-E10, Logitech G Pro X Headset, Rode NT-USB Mini, LG UltraWide 34WN80C, Apple Magic Keyboard, That, Parent, Partner, Although, Company, Half, Natural, Method, Rich, See, May, Point, Into, Maintain, Yourself, Performance, Suddenly, Final, Heart, Fish, Hundred, Factor, Apple iPhone 14, Samsung Galaxy S22, Lenovo ThinkPad X1, Dell UltraSharp 27", Logitech MX Master 3, Apple MacBook Pro 16", HP LaserJet Pro M404n, Canon EOS R6, TP-Link Archer AX55, Philips Hue Starter Kit, Razer DeathAdder V2, Corsair K95 RGB Platinum, Sony PlayStation 5, Xbox Series X, Apple iPad Air 5, Amazon Kindle Paperwhite, AMD Ryzen 9 7950X, Intel Core i9-13900K, ASUS ZenBook 14 OLED, Synology DS920+.

### 3.9.2. Experiment 2 – nejjobsazenější pozice

V druhém experimentu měl chatbot odpovědět na prompt „Jaká je nejjobsazenější pozice a kolik je na ní krabic. Odpověz jednou větou.,, Správná odpověď podle dat byla pozice NDHZ682 s počtem 23 krabic. V tomto experimentu byly všechny odpovědi správné. Důvodem může být jednodušší dotaz s kratší odpovědí. Součástí experimentu byla i přímá kontrola správnosti, kde se kontrolovalo, zda odpověď obsahuje správný název a množství, které bylo v daný okamžik získáno z databáze.

V následující tabulce je ukázka prvních 5 odpovědí.

Tabulka 5 Ukázka výsledku druhého experimentu – nejjobsazenější pozice Zdroj: vlastní

run	actual	is correct
1	Nejobsazenější pozice je "NDHZ682" s 23 krabicemi.	TRUE
2	Nejobsazenější pozice je "NDHZ682" s 23 krabicemi.	TRUE
3	Nejobsazenější pozice je "NDHZ682" s 23 krabicemi.	TRUE
4	Nejobsazenější pozice je "NDHZ682" s 23 krabicemi.	TRUE
5	Nejobsazenější pozice je "NDHZ682" s 23 krabicemi.	TRUE

### 3.9.3. Pozorování z implementace a z experimentů

Během vývoje a testování systému s integrovaným řešením umělé inteligence se objevily opakující se problémy, které zásadním způsobem ovlivňují spolehlivost odpovědí generovaných chatbotem. Jedním z nejčastějších nedostatků byla odpověď indikující nedostupnost dat, přestože backend systému poskytoval všechny potřebné informace. Příčinou této chyby bylo opakované dotazování se na nesprávné metody nebo nedostatečně specifikované promptování. V rámci běžné konverzace bylo možné tyto problémy částečně eliminovat zpřesňováním dotazů nebo opakováním požadavku. Avšak v oblasti automaticky generovaných statistik tato možnost obvykle neexistuje, a chybovost tak byla výrazně vyšší.

Z experimentálního testování vyplynulo, že chatbot při opakovaném dotazu často nerevaliduje data, ale vrací dříve vygenerovanou odpověď. Tento jev byl pozorován i při manuálním testování, po prvním dotazu si model interně uchoval výsledek a v dalších případech již nedocházelo ke kontaktu s databází. Takové chování může být problematické zejména ve skladových systémech, kde se data často mění (např. přibývají a ubývají krabice na skladových pozicích).

Další významné omezení představuje **finanční náročnost provozu**. Z praxe vyplývá, že jedno vykreslení 4–5 statistických grafů může vyjít přibližně na 10 dolarů, a obdobné náklady mohou vzniknout při sérii 100 jednoduchých textových promptů. V prostředí menších firem by takový model nebyl dlouhodobě udržitelný bez zavedení systému kreditního nebo předplaceného přístupu. Frekventovaný provoz ve skladu, vyžadující časté překreslování statistik a opakované dotazování, by mohl generovat neúnosné náklady.

Možným řešením je zavedení měsíčního rozpočtu (budgetu) pro jednotlivé klienty, přičemž po jeho vyčerpání by klient měl možnost si přístup k dalším dotazům připlatit. Tato varianta by však vyžadovala vytvoření samostatného OpenAI projektu (úctu) pro každého klienta, což by přineslo významné technické komplikace, zejména nutnost individuálního nastavení a správy každého připojení. Aktuální implementace totiž podporuje pouze rozlišení vláken podle uživatelů a klientů v rámci jednoho centrálního OpenAI účtu, což je nedostačující v případě vícenásobného komerčního nasazení.

V rámci testování byla rovněž zvažována možnost využití alternativního poskytovatele služeb umělé inteligence, konkrétně Azure OpenAI od společnosti Microsoft. Toto řešení však v současnosti vykazuje přílišná omezení, zejména v oblasti volání vlastních metod a integrace backendových dat, a tudíž bylo pro potřeby tohoto projektu shledáno jako nevyhovující.

## 4. Závěr

Diplomová práce se zaměřila na návrh a implementaci webové aplikace StockWise, která propojuje moderní webové technologie s prvky umělé inteligence za účelem efektivnější správy skladových procesů. Výsledný systém postavený na Django, Next.js a Supabase obsahuje inteligentního chatbota založeného na OpenAI, jenž umožňuje správu skladu přirozeným jazykem, a rovněž modul pro generování statistických přehledů.

V praktické části byly ověřeny nejen funkční schopnosti systému, ale také samotná efektivita využití AI. Experimenty prokázaly, že chatbot je dobře využitelný pro jednoduché úkony, jako je vytváření, úprava a mazání dat, především v hromadné podobě. Při komplexnějších požadavcích se však objevily nedostatky v přesnosti odpovědí, zejména při dotazech na úplnost dat. Závažné se ukázalo i chování modelu, kdy často vracel odpovědi z cache místo aktuálních dat, což může vést k chybným rozhodnutím. Významným limitem je rovněž finanční náročnost, generování několika statistik či série promptů může znamenat desítky dolarů nákladů, což omezuje škálovatelnost a udržitelnost řešení bez dodatečných opatření.

Z těchto důvodů se jako nejvhodnější cesta jeví nasazení základního chatbota pro rutinní operace, zatímco pokročilé AI statistiky je třeba zatím považovat za doplňkovou funkci vhodnou spíše pro větší firmy s vyššími rozpočty. Do budoucna může být systém rozvíjen směrem k větší spolehlivosti například optimalizací promptů, zpřesněním dotazování a rozšířením chatbota o kontextové odpovědi. Významným krokem by mohl být také vývoj mobilní aplikace pro skladníky, která by umožnila přímou interakci s chatbotem v provozu. Provozní náklady lze částečně řešit zavedením měsíčních rozpočtů na AI dotazy.

Závěrem lze konstatovat, že i přes současná omezení představuje využití umělé inteligence v oblasti skladového hospodářství slibnou cestu, která při správném a kontrolovaném nasazení může zefektivnit řadu procesů a přinést firmám skutečný přínos.

## 5. Použitá literatura

- 1 HORÁKOVÁ, Helena a KUBÁT, Jiří. *Řízení zásob: logistické pojetí, metody, aplikace, praktické úlohy*. 3., přeprac. vyd. Praha: Profess Consulting, 1999. ISBN 80-85235-55-2.
- 2 DRAHOTSKÝ, Ivo a ŘEZNÍČEK, Bohumil. *Logistika: procesy a jejich řízení*. Brno: Computer Press, 2003. ISBN 80-7226-521-0. ENHOLM, Ida Merete; PAPAGIANNIDIS, Emmanouil
- 3 CHOPRA, Sunil a Peter MEINDL, 2016. *Supply Chain Management: Strategy, Planning, and Operation*. 6. Boston: Pearson. ISBN ISBN 80-85235-55-2.
- 4 DONALD J., BOWERSOX, CLOSS DAVID J. a COOPER M. BIXBY, 2012. *Supply Chain Logistics Management*. 4. New York: McGraw-Hill. ISBN 978-0078024054.
- 5 JIRSÁK, Petr, Felipe MARTÍNEZ a Petr PERNICA, 2023. *Vybrané kapitoly z přepravy a logistiky V (cesta k chytré logistice)* [online]. Vysoká škola ekonomická v Praze [cit. 2025-04-04]. ISBN 978-80-245-2490-0. Dostupné z: <https://oeconomica.vse.cz/wp-content/uploads/publication/24009/Vybrane-kapitoly-z-prepravy-a-logistiky-V.pdf>
- 6 VEBER, Jaromír. *Digitalizace ekonomiky a společnosti: výhody, rizika, příležitosti*. V nakladatelství Management Press vydání 1. Praha: Management Press, 2018. ISBN 978-80-7261-554-4.
- 7 SEDLÁČEK, Miroslav a STŘELEČEK, Tomáš. *Civilní právo a nové technologie*. Právní monografie. Praha: Wolters Kluwer, 2022. ISBN 978-80-7676-503-0.
- 8 MIKALEF, Patrick a KROGSTIE, John. Artificial Intelligence and Business Value: a Literature Review. Online. *Information systems frontiers*. 2022, roč. 24, č. 5, s. 1709-1734. ISSN 1387-3326. Dostupné z: <https://doi.org/10.1007/s10796-021-10186-w>. [cit. 2024-09-26].
- 9 TIŠNOVSKÝ, Pavel. *Evoluce Pythonu: moderní techniky, nová syntaxe a trendy*. CZ.NIC. Praha: CZ.NIC, z.s.p.o., 2024. ISBN 978-80-88168-76-8.
- 10 SUMMERFIELD, Mark a KREJČÍ, Lukáš. *Python 3: výukový kurz*. 2. vydání. Brno: Computer Press, 2021. ISBN 978-80-251-5030-6.
- 11 PECINOVSKÝ, Rudolf. *Python: kompletní příručka jazyka pro verzi 3.10*. Online. Myslíme v-. Praha: Grada Publishing, 2021. ISBN 978-80-271-3442-7. Dostupné z: <https://www.bookport.cz/kniha/python-10436/>. [cit. 2025-04-02].
- 12 CHOLLET, François. *Deep learning v jazyku Python: knihovny Keras, Tensorflow*. Online. Knihovna programátora. Praha: Grada Publishing, 2019. ISBN 978-80-247-3100-1. Dostupné z: <https://www.bookport.cz/kniha/deep-learning-v-jazyku-python-6045/>. [cit. 2025-04-02].
- 13 COLAS, Thibaud a Adrian HOLOVATY, 2024. Django Documentation. *Django Documentation* [online]. [cit. 2025-04-02]. Dostupné z: <https://docs.djangoproject.com/en/5.2/>
- 14 CLARK, Andrew a Dan ABRAMOV, 2013. React. *React* [online]. [cit. 2025-04-02]. Dostupné z: <https://react.dev>
- 15 VERCEL, INC., 2025. Vercel. VERCEL, INC. *Vercel* [online]. [cit. 2025-04-03]. Dostupné z: <https://nextjs.org/docs>

- 16 KUBÁLE, Ondřej, 2016. Optimalizace skladového hospodářství ve výrobním podniku KOH-I-NOOR Mladá Vožice a.s. Českých Budějovicích. Diplomová práce. Vysoká škola technická a ekonomická v Českých Budějovicích.
- 17 MULAČOVÁ, V., P. MULAČ, 2013. Obchodní podnikání ve 21. století. 1. vyd. Praha: Grada Publishing a.s. ISBN 978-80-247-4780-4.
- 18 LUKOSZOVA, Xenie, 2012. Logistické technologie v dodavatelském řetězci. 1. vyd. Praha: Ekopress ISBN 978-80-86929-89-7.
- 19 SYNEK, M., E. KISLINGEROVÁ., 2010. Podniková ekonomika. 5., přeprac. a dopl. vyd. Praha: C.H. Beck ISBN 978-80-7400-336-3.
- 20 JAKUBÍKOVÁ, Dagmar, 2008. Strategický marketing. 1. vyd. Praha: Grada Publishing a.s. ISBN 978-80-247-2690-8.
- 21 LAMBERT, D., M., J. R. STOCK a L. M. ELLRAM., 2005. Logistika: příkladové studie, řízení zásob, přeprava a skladování, balení zboží. 2. vyd. Brno: CP Books ISBN 80-251-0504-0.
- 22 GOPALAKRISHNAN, P., A. HALEEM, 2015. Handbook of Materials Management. 2 edition. Delhi: PHI Learning ISBN 9788120348011.
- 23 TOP 7 TRENDŮ V LOGISTICE A SKLADOVÁNÍ NA ROK 2024: JAK AI TRANSFORMUJE SKLADOVOU LOGISTIKU, 2023. ANASOFT [online]. (1) [cit. 2025-04-16]. Dostupné z: [https://www.anasoft.com/emans/cz/home/Novinky-blog/Blog/top-7-trendu-v-logistice-2024-AI-sklad?utm\\_source=chatgpt.com](https://www.anasoft.com/emans/cz/home/Novinky-blog/Blog/top-7-trendu-v-logistice-2024-AI-sklad?utm_source=chatgpt.com)
- 24 KADLAS BLÜMELOVÁ, Kristina, 2023. Umělá inteligence od základu mění logistiku i skladové hospodářství. Technický týdeník [online]. (1) [cit. 2025-04-16]. Dostupné z: [https://www.technickytydenik.cz/rubriky/poutaky/umela-inteligence-od-zakladu-meni-logistiku-i-skladove-hospodarstvi\\_59024.html](https://www.technickytydenik.cz/rubriky/poutaky/umela-inteligence-od-zakladu-meni-logistiku-i-skladove-hospodarstvi_59024.html)
- 25 LÁSKOVÁ, Romana, 2024. Umělá inteligence ve skladových procesech. NEDCON SALES [online]. (1) [cit. 2025-04-16]. Dostupné z: [https://nedconsales.cz/blog/umela-inteligence-ve-skladovych-procesech/?utm\\_source=chatgpt.com](https://nedconsales.cz/blog/umela-inteligence-ve-skladovych-procesech/?utm_source=chatgpt.com)
- 26 CHRISTOPHER, Martin. *Logistika a řízení zásob*. 4. vyd. Praha: Computer Press, 2016.
- 27 CEMPÍREK, Václav. *Logistika: metody používané pro zefektivnění logistických systémů*. 1. vyd. Pardubice: Univerzita Pardubice, 2007. s. 55.
- 28 LANGEROVÁ, Dana. *Outsourcing logistických služeb*. Praha: Oeconomica, 2014.
- 29 LANE, Dale. *Strojové učení pro děti*. 1. vyd. Brno: Computer Press, 2020. ISBN 978-80-251-5154-9.
- 30 KREJČÍ, Michal. *Modely dynamického chování robota*. Bakalářská práce. Plzeň: Západočeská univerzita v Plzni, Fakulta aplikovaných věd, 2015. Dostupné z: <https://theses.cz/id/0mp9v3/STAG95176.pdf>
- 31 KAŠPÁREK, Michal. GPT a další velké jazykové modely [online]. 2023 [cit. 19. 4. 2025]. Dostupné z: <https://michalkasparek.cz/llms.html>
- 32 ČESKÁ JUSTICE. Jak funguje jazykový model pro AI? Obsahuje všechny knihy, které byly napsány, i ty, které napsány nebyly [online]. 2023 [cit. 19. 4. 2025]. Dostupné z: <https://www.ceska-justice.cz/2023/05/jak-funguje-jazykovy-model-pro-ai-obsahuje-vsechny-knihy-ktere-byly-napsany-i-ty-ktere-napsany-nebyly/>

- 33 ROOT.CZ. AI v praxi: Jak efektivně integrovat umělou inteligenci do vývoje a firemních procesů [online]. 2023 [cit. 19. 4. 2025]. Dostupné z: <https://www.root.cz/pr-clanky/ai-v-praxi-jak-efektivne-integrovat-umelou-inteligenci-do-vyvoje-a-firemnich-procesu/>
- 34 BUDILOVÁ, Eva. *Analýza možností využití umělé inteligence v oblasti human resources*. Bakalářská práce. České Budějovice: Jihočeská univerzita v Českých Budějovicích, Ekonomická fakulta, 2023. Dostupné z: <https://wstag.jcu.cz/...> (přímý link je příliš dlouhý – doporučuji použít trvalý odkaz ze systému STAG)
- 35 SAP. Warehouse Automation – SAP Extended Warehouse Management [online]. SAP, 2024 [cit. 19. 4. 2025]. Dostupné z: <https://www.sap.com/products/scm/extended-warehouse-management/warehouse-automation.html>
- 36 HOUGHTON, Tom. Amazon's warehouse robots cut costs by 25% and are getting more capable. *Financial Times* [online]. 2023 [cit. 19. 4. 2025]. Dostupné z: <https://www.ft.com/content/31ec6a78-97cf-47a2-b229-d63c44b81073>
- 37 Čárový kód už má více než 40 let, 2025. *DATASCAN* [online]. [cit. 2025-05-04]. Dostupné z: <https://www.datascan.cz/blog/carovy-kod-uz-ma-vice-nez-40-let>
- 38 Meet Proteus, 2024. *Amazon* [online]. [cit. 2025-05-04]. Dostupné z: <https://www.aboutamazon.com/stories/amazon-robotics-autonomous-robot-proteus-warehouse-packages>