VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO EVIDENCI VÝROBY A SKLADU

PRODUCTION AND STOCK INFORMATION SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — DOMINIK TOMÁŠIK
AUTHOR

VEDOUCÍ PRÁCE — Ing. RADEK BURGET, Ph.D.
SUPERVISOR

BRNO 2019

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

21437

Student: **Tomášik Dominik**
Program: Informační technologie
Název: **Systém pro evidenci výroby a skladu**
Production and Stock Information System
Kategorie: Informační systémy
Zadání:

1. Prostudujte současné technologie pro tvorbu informačních systémů s webovým rozhraním.
2. Seznamte se s požadavky zadavatele na informační systém výroby a skladu a prozkoumejte existující systémy s podobným zaměřením.
3. Analyzujte požadavky a navrhněte architekturu systému. Uvažujte možnost zpětného dohledání veškerých akcí uživatelů.
4. Po dohodě s vedoucím implementujte navržený systém pomocí vhodných technologií. Implementujte možnost exportu tiskových sestav a statistik ve formátu XLSX a PDF.
5. Proveďte testování systému v reálném prostředí.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Gutmans, A., Rethans, D., Bakken, S.: Mistrovství v PHP 5, Computer Press, 2012
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Burget Radek, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 30. října 2018

Abstrakt

Táto bakalárska práca sa zaoberá analýzou, návrhom a implementáciou informačného systému pre evidenciu výroby a skladu vo forme webovej aplikácie. Hlavnou úlohou takéhoto informačného systému je ušetriť čas, jak obchodným manažérom s predajom produktov, tak správcom skladu a majitelovi s evidenciou. Pre zaistenie danej úlohy, systém umožňuje evidenciu predajov, klientov, dodávateľov, zásob a produktov. Pri každej kategórií v evidenciách existuje štatistika, ktorú je možné zoradiť, prípadne filtrovať podľa rôznych špecifikácií a majiteľ, tak môže všetko ľahko sledovať. Systém je obzvlášť výhodný pre obchodných manažérov pri predaji, keďže jediné čo zadávajú po zadaní klienta a produktu je množstvo predaných produktov, položky ako cena, a zľava produktu sa načítavajú automaticky. Navrhnutý systém je implementovaný technológiami HTML, CSS, MySQL, jQuery, AJAX a PHP. Výsledná webová aplikácia má slúžiť najmä pre spoločnosti, ktoré si vyrábajú vlastné produkty a následne ich predávajú svojím klientom.

Abstract

This bachelor's thesis deals with the analysis, design, and implementation of an information system for managing production and stock in the form of a web application. The main task of this information system is to help the manager save time with managing and to help salespeople with selling products. The mentioned task is ensured because the system is capable of managing sales, clients, suppliers, supplies and products. All categories have statistics which can be ordered or filtered and because of that, the owner can easily observe everything. The system is especially useful for salespeople when they are selling products only thing they enter after choosing a client and product is an amount because price and discount are automatically loaded. The designed system is implemented with technologies like HTML, CSS, MySQL, jQuery, AJAX, and PHP. The web application is mainly for companies that manufacture own products and then sell them to their clients.

Kľúčové slová

Informačný systém, webová aplikácia, evidencia skladu, evidencia výroby, evidencia predajov, Bootstrap, PHP, MySQL, AJAX, jQuery, DataTables

Keywords

Information system, web application, inventory management, production management, sales management, Bootstrap, PHP, MySQL, AJAX, jQuery, DataTables

Citácia

TOMÁŠIK, Dominik. *Systém pro evidenci výroby a skladu.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

Systém pro evidenci výroby a skladu

Prehlásenie

Prehlasujem, že som túto prácu vypracoval samostatne pod vedením pána Ing. Radka Burgeta Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

. . . . . .
Dominik Tomášik
7. mája 2019

Poďakovanie

Chcel by som sa poďakovať svojmu vedúcemu, Ing. Radkovi Burgetovi, Ph.D, za pomoc pri riešení tejto bakalárkej práce. Rovnako by som sa chcel poďakovať zamestnancom spoločnosti Broader s.r.o. za pomoc s testovaním systému.

Obsah

Kapitola 1

Úvod

Spolu s vývinom informačných technológií dochádza aj k využívaniu informačných systémov v podnikoch. Je len otázkou času, kedy každý podnik či už malý alebo veľký bude využívať nejaký informačný systém.

Podniky, ktoré ešte nemajú informačný systém a evidujú všetko na papier, strácajú čas, nemajú prehľad o dátach v reálnom čase a hlavne dochádza k menšej produktivite. Práve preto pre každý podnik je výhodne mať informačný systém. Kvalitný informačný systém umožňuje jednoduchú evidenciu, efektívnosť, ušetrí čas a hlavne zvýši produktivitu.

V dnešnej dobe už existuje mnoho zadarmo dostupnúch informačných systémov, ktoré zahrňujú skladovú evidenciu buď s predajom produktov alebo bez, ale žiadny taký systém neobsahuje proces výroby produktu spolu s jej evidenciou.

Cieľom tejto práce je navrhnúť a implementovať práve taký informačný systém, ktorý uľahčí prácu zamestnancom a manažérom s evidenciou. Narozdiel od ostatných informačných systémov, tento bude evidovať nielen predaje a sklad, ale aj úplný proces výroby produktov.

Táto práca bola pravidelne konzultovaná so spoločnosťou Broader s.r.o.[1], čo pomohlo pri analýze a testovaní systému. Spoločnosť Broader s.r.o. si vyrába vlastné produkty, ktoré následne uskladňuje do vlastného skladu a neskôr ich predáva klientom. Práve preto je pre nich tento informačný systém vhodný.

Text práce je štruktorovaný do niekoľkých kapitol, ktoré ju popisujú. V kapitole 2 sú špecifikované a následne podrobne analyzované požiadavky na informačný systém, ktoré boli zadané zadaváteľom. Technológie, ako jazyky a diagramy, ktoré boli použité na návrh a implementáciu systému sú popísané v kapitole 3. Na základe analýzy požiadavkov je v kapitole 4 popísaný návrh systému. V kapitole 5 sa nachádza popis implementácie informačného systému. Obsah kapitoly 6 tvorí priebeh a výsledky testovania systému.

[1] Broader s.r.o. https://broader.sk/

Kapitola 2

Špecifikácia a analýza požiadavkov

Každý informačný systém v dnešnej dobe potrebuje dobre špecifikované požiadavky od zákazníka. Pre dobre funkčný informačný systém je nutné zistiť od zákazníka jeho potreby a čo očakáva od funkčného systému. Výsledné špecifikácie je vždy dobre prejsť so zákazníkom ešte raz, aby nedošlo k nejakému nedorozumeniu. Po potvrdení výsledných špecifikácií zákazníkom nastáva ich analýza. Na základe požiadavkov, bol systém rozdelený do štyroch častí:

- Používatelia systému
- Evidencia skladu
- Evidencia výroby
- Evidencia predajov

V tejto kapitole sú spísané požiadavky na informačný systém, ktoré už boli analyzované. Prezentačná časť webovej stránky, respektíve to čo by mala webová stránka splňovať z hľadiska výzoru je v sekcii 2.1. V sekciách 2.2 až 2.5 sú popísané jednotlivé časti systému.

2.1 Prezentačná časť webového rozhrania

Táto časť je zameraná najmä na celkový dizajn, prezentačná časť jednotlivých častí systému je popísaná u samotných sekciách. Zákazník vyžadoval aby web mal moderný dizajn v ktorom sa bude dať ľahko orientovať, aby bolo možné pristupovať na web z hocijakého zariadenia s internetom. Čo sa týka tabuliek, ktoré budú zobrazovať údaje, tak by malo byť schopné v nich vyhľadávať a následné údaje exportovať.

2.2 Používatelia informačného systému

V rámci používateľov informačného systému, môžu existovať 3 role používateľov:

- Administrátor
- Správca skladu
- Predajca

Nových používateľov do systému môže pridávať len administrátor. Pri prihlasovaní používateľ by mal mať možnosť aby si systém zapamätal prihlásenie, tak aby pri ďalšej návšteve nemusel znova zadávať údaje. Administrátor pri pridávaní zadáva prihlasovacie meno, heslo, rolu, telefón, e-mail, pričom všetky sú povinné polia. Pridaný používateľ sa môže prihlásiť do systému. Každý používateľ môže upraviť len svoj profil, pozrieť si len svoje štatistiky a taktiež sa môže odhlásiť zo systému. Takýchto používateľov môže len administrátor upravovať alebo vymazať. Okrem toho administrátor môže meniť hlavné nastavenia, v ktorých je meno, adresa, ičo, dič, ič-dph spoločnosti a tiež môže meniť dph. Administrátor má ešte ako jediný prístup k záznamom, kde sa zaznamenáva každá činnosť čo sa vykoná v systéme, ktorú si môže filtrovať podľa dátumu a času, používateľa alebo kategórií.

Každý používateľ má minimálne práva predajcu, takže má prístup k nasledujúcim kategóriám:

- Sklad zásob
- Sklad výrobkov
- Predaje
- Klienti
- Úlohy

Správca skladu na rozdiel od predajcu má navyše prístup k:

- Kategóriam
- Dodávateľom
- Kategóriam produktov
- Produktom
- Typom obchodov
- Statusom klientov

Administrátor má práva ku všetkému, to znamená, že má ešte prístup k:

- Hlavným nastaveniam
- Úloham
- Záznamom
- Používateľom
- Štatistikám

Vyššie uvedené práva rôznych rolí sú viac popísane v nasledujúcich sekciách.

2.2.1 Úlohy

Len administrátor je schopný vytvoriť úlohu, pri vytváraní vyberá, ktorému používateľovi bude pridelená úloha, dátum do kedy by mal priradený používateľ úlohu dokončiť, jej popis a prioritu úlohy. Priorita môže byť nízka, stredná alebo vysoká. Po pridaní úlohy sa danému používateľovi vizuálne zobrazí.

Pridané úlohy, môže označiť za splnené alebo nesplnené len administrátor, okrem toho ich môže upraviť, alebo vymazať.

Každý používateľ má prístup k svojim úlohám v tabuľke, v ktorej môže filtrovať podľa statusu (nesplnená, splnená) a podľa priority.

2.2.2 Štatistika

Štatistika sa týka najmä štatistiky používateľov informačného systému. Podobne ako pri úlohách, sem má prístup len administrátor. Administrátor si môže zobraziť štatistiku hocijakého používateľa.

Táto štatistika sa delí na dve časti, štatistika klientov a predajov. Štatistika klientov zobrazí počet pridaných klientov za posledných sedem dní, tridsať dní, za každý mesiac v poslednom roku a tiež za celé obdobie od registrácie používateľa. Štatistika predajov zobrazuje údaje rovnakým štýlom ako štatistika klientov, ale zameriava sa najmä na počet predajov, predané množstvo a tržbu z predajov. Pri oboch štatistíkách sa musí zobraziť graf, ktorý tie údaje zobrazí graficky.

2.3 Požiadavky na evidenciu skladu

V tejto sekcii sú popísané veci spájajúce sa s evidenciou skladu. Tvoria ju tri hlavné časti:

1. **Dodávatelia** - od ktorých sa objednávajú zásoby do skladu
2. **Členenie zásob** - ide o kategorizovanie zásob
3. **Sklad zásob** - poskytuje aktuálny prehľad zásob, taktiež umožňuje vytvorenie a spravovanie objednávok

Ich podrobnejší popis je v podkapitolách 2.3.1 až 2.3.3.

2.3.1 Dodávatelia

Administrátor a správca skladu majú právo vytvoriť dodávateľa. Pri vytváraní dodávateľa sa zadávajú identifikačné a kontaktné údaje. Identifikačné údaje tvoria meno firmy, adresa, ičo, dič a ič-dph, pričom sú všetky povinné polia okrem ič-dph. Kontaktné údaje tvoria celé meno, telefón, email, web a poznámka. Z kontaktných údajov sú povinné polia len celé meno a telefón. Pridaných dodávateľov môžu jak administrátor, tak správca skladu upraviť poprípade vymazať. Dodávateľ slúži na to aby sa vedelo od koho sa objednávajú zásoby do skladu.

Obe role majú prístup k tabuľke zoznamu dodávateľov, kde vidia všetky ich údaje, ktoré môžu filtrovať podľa zadaného reťazca a môžu ich odtiaľ upraviť alebo vymazať.

2.3.2 Členenie zásob

Rovnako ako pri dodávateľoch sem má prístup používateľ s právami administrátora alebo správcu skladu. Členenie zásob má tri častí:

1. **Kategórie** - Slúžia na kategorizovanie zásoby. Používateľ pri vytváraní zadáva len meno. Môže to byť napríklad: surovina, tekutý materiál, pevný materiál alebo doplnkový materiál.

2. **Podkategórie** - Pri vytváraní podkategórie používateľ musí vybrať kategóriu do ktorej bude patriť podkategória a tiež zadáva jej meno. Podkategória môže byť: jablko, hruška alebo hrozno patriace do kategórie Surovina.

3. **Položky** - Definuje konkrétnu odrodu danej zásoby, napríklad pri podkategórií jablko to je Jonathan alebo Golden Delicious.

Okrem vytvárania, má používateľ možnosť ich upraviť alebo vymazať. Používateľ ma taktiež prístup k ich zoznamu, kde si ich môže filtrovať či už podľa dátumu pridania, kategórie, podkategórie alebo zadaného reťazca.

2.3.3 Sklad zásob

Sklad zásob sa skladá z troch častí:

1. **Prehľad zásob** - Slúži na zobrazenie aktuálneho stavu skladu zásob, ktorý si môže pozrieť každý používateľ systému. Je v ňom možná filtrácia podľa kategórií, podkategórií, suroviny alebo množstva.

2. **Vytvorenie novej objednávky** - Používateľ s právami administrátora alebo správcu skladu, môže vytvárať takúto objednávku. Pri vytváraní objednávky používateľ vyberá dodávateľa spomenutého v podkapitole 2.3.1, kategóriu, podkategóriu a konkrétnu položku, ďalej zadáva množstvo a cenu. V jednej objednávke je možné objednať viac zásob naraz.

   Ak sa objednáva len jedna zásoba, je nutné vypísať sumu s DPH a bez za danú položku, a tiež samotné DPH. Pri objednávke kde je viac zásob musí vypísať nielen to čo pri jednej zásobe, ale aj sumu všetkých zásob s DPH aj bez, a rovnako aj sumu všetkých DPH.

   Po vytvorení objednávky sa danej objednávke automaticky nastaví stav objednávky „neschválený“. Stav objednávky môže byť buď schválený alebo neschválený. V tomto prípade schválená znamená, že je objednávka už naskladnená na sklade. Tento stav, respektíve naskladnenie objednávky môže nastaviť jak administrátor, tak správca skladu.

3. **Zoznam objednávok** - Zobraziť zoznam objednávok môže zobraziť len administrátor a správca skladu. V zozname je možné filtrovať podľa toho kto vytvoril objednávku, dátumu, stavu objednávky a vyhľadávať podľa reťazca. Vyfiltrované výsledky je možné exportovať vo formátoch .xlsx, .csv alebo .pdf.

2.4 Požiadavky na evidenciu výroby

Obsahom tejto sekcie sú analyzované požiadavky na evidenciu výroby a jej proces. Podobne ako sklad zásob je zložená z troch častí:

1. **Kategórie produktov** - slúži na kategorizovanie produktov.
2. **Výroba produktu** - ide o výrobný proces produktu.
3. **Sklad výrobkov** - umožňuje aktuálny prehľad produktov na sklade.

Ich podrobnejší popis je v podkapitolách 2.4.1 až 2.4.3.

2.4.1 Kategórie produktov

Prístup do tejto časti má len používateľ s právami administrátora alebo správcu skladu. Kategórie produktov slúžia na členenie jednotlivých produktov s rôznym dátumom spotreby. Pri vytváraní sa zadáva len meno kategórie, môže to byť napríklad Jablko, sklo 0,75 l.

Používateľ má taktiež prístup k zoznamu kategórií produktov, kde má možnosť ich upraviť alebo vymazať.

2.4.2 Výroba produktu

Vytvoriť nový produkt môže používateľ s právami administrátora alebo správcu skladu. Pri vytváraní sa zadávajú niekoľko údajov, rozdelených na päť časti:

1. **Základné údaje** - Používateľ vyberá kategóriu produktu do ktorej bude nový produkt patriť. Ďalej zadáva dátum výroby, trvanie výroby v hodinách, šaržu, minimálnu trvanlivosť, vyrobené množstvo a predajnú cenu bez DPH.
2. **Náklady** - Používateľ zadáva spotrebovanú elektrinu, vodu a plyn v odpovedajúcich metrických jednotkách spolu s cenou.
3. **Použité materiály** - Sú to zásoby, ktoré sa použili pri výrobe produktu a už boli naskladnené, čo znamená, že sú na sklade. Používateľ vyberá kategóriu následne odpovedajúcu podkategóriu a nakoniec konkrétnu položku na sklade. Taktiež zadáva použité množstvo a cenu bez DPH za kus. Takýchto použitých materiálov môže byť pri výrobe produktu viac ako jedna.
4. **Zamestnanci pri výrobe** - Ide o zamestnancov, ktorí sa podielali na výrobe nového produktu. Používateľ zadáva celé meno zamestnanca, počet odpracovaných hodín a mzdu bez DPH. Pri výrobe môže byť počet zamestnancov aj viac ako jeden.
5. **Ostatné náklady** - Jedná sa o náklady, ktoré nemusia byť predvídateľné, ako nejaká porucha alebo výpadok elektriny počas výroby. Zadáva sa popis nákladu a cena bez DPH.

Pri nákladoch, materiáloch, zamestnancoch a ostatných nákladoch sa musí vypisovať suma s DPH, suma bez DPH a samotné DPH pri každej položke rovnako ako suma jednotlivých položiek. Okrem toho sa musí vypisovať celková suma všetkého spolu jak s DPH, tak bez.

Používateľ musí mať prístup k zoznamu takýchto produktov, kde ich môže upravovať a vymazať. Ďalej musí byť dostupná tlač vyrobeného produktu so všetkými informáciami o danom produkte.

2.4.3 Sklad výrobkov

Sklad výrobkov je dostupný pre každého používateľa informačného systému. Poskytuje, ktoré produkty sa nachádzajú práve na sklade v akom množstve, aká bola zadaná predajná cena s dátumom spotreby. Taktiež je tu možná filtrácia podľa dátumu spotreby, kategórie produktu alebo podľa zadaného reťazca, výsledky je možné exportovať vo formátoch .xlsx, .csv a .pdf.

2.5 Požiadavky evidenciu predajov

Táto sekcia popisuje zákazníkové požiadavky spojené s evidenciou predajov. Po analýze bola evidencia predajov rozdelená na štyri časti:

1. **Typy obchodov** - definujú obchody, do ktorých klienti patria.
2. **Status klientov** - definuje rozdielné statusy priradené klientom.
3. **Klienti** - spravovanie klientov, ktorým sa predávajú produkty.
4. **Predaj produktov** - ide o celkový proces predaja.

Konkrétny popis vyššie spomenutých časti je v podkapitolách 2.5.1 až 2.5.4.

2.5.1 Typy obchodov

Každý klient musí patriť do nejakého typu obchodu. Typ obchodu slúži na definovanie percentuálnej zľavy na jednotlivé kategórie produktov spomenuté v podkapitole 2.4.1. Typ obchodu môže vytvoriť len používateľ s právami administrátora alebo správcu skladu, kde zadáva len jeho meno. Po vytvorení môže používateľ priradiť percentuálnu zľavu kategóriám produktov.

Ďalej musí byť pre používateľa dostupná tabuľka, ktorá musí zobrazovať všetky typy obchodov, kde sa bude dať upraviť meno a ich zľavy pre kategórie produktov, a taktiež možnosť ich vymazania.

2.5.2 Status klientov

Status klientov slúži na priradenie statusov klientom, ako napríklad aktívny, neaktívny, zrušený a iné. Takéto statusy môže vytvárať používateľ s právami administrátora alebo správcu skladu. Pri vytváraní zadáva dve polia, ktoré sú obe povinné, prvý údaj je meno statusu a druhý je farba statusu. V zozname statusov je možné vytvorené statusy upravovať alebo ich vymazať.

2.5.3 Klienti

Klientov môže pridávať každý používateľ informačného systému. Klienti sa delia na súkromnú osobu a firemného klienta. Pri vytváraní klienta používateľ vyberá či ide o súkromnú osobu alebo firemného klienta a ďalej mu priradí typ obchodu. Ak ide o súkromnú osobu, používateľ zadáva ešte celé meno klienta, telefónne číslo a e-mail, pričom povinný údaj je len jeho meno. Ak pôjde o firemného klienta zadáva názov firmy, kontaktnú osobu, adresu prevádzky, ičo, dič, ič-dph, telefónne číslo a e-mail, povinné údaje v tomto prípade sú

názov firmy, adresa prevádzky, ičo a dič. Každému klientovi, ktorý bol pridaný je priradený prednastavený status „aktívny“.

Každý používateľ má prístup k zoznamu svojich klientov, kde si môže pozrieť v tabuľke všetky informácie o klientoch, ktorých pridal. Okrem informácií, ktoré zadával pri vytváraní klienta sa u každého klienta zobrazí koľkokrát si objednával produkty. Používateľ môže svojich klientov upravovať alebo vymazať. Pri upravovaní môže používateľ klientovi nastaviť status, spomenutý v podkapitole 2.5.2. V tomto zozname je možné filtrovať podľa dátumu pridania, typu obchodu, typu klienta, statusu alebo zadaného reťazca.

Administrátorská sekcia v rámci klientov, rozširuje pôvodnú tabuľku. Administrátor vidí všetkých klientov a tiež kto daného klienta spravuje, ktorých môže upraviť alebo vymazať. Ďalej môže nastať prípad, keď zamestnanec odíde a je nutné presunúť všetkých klientov na iného zamestnanca. Tento prípad rieši administrátorská sekcia, kde môže presunúť klientov od nejakého používateľa buď po jednom alebo všetkých naraz na iného používateľa. Oproti normálnemu zoznamu, tu je možné filtrovať aj podľa toho, kto spravuje klienta.

2.5.4 Predaj produktov

Každý používateľ po pridaní klientov má možnosť im predať vyrobené produkty. Pri vytváraní objednávky používateľ vyberá klienta, ktorému sa predajú produkty a zadáva ešte dátum predaja. Po vybratí klienta, používateľ musí špecifikovať aké produkty chce predať klientovi. Ako prvé musí vybrať kategóriu produktov a následne vyberie samotný produkt. Cena produktu by sa mala načítať podľa predajnej ceny, ktorá sa zadávala pri vytváraní produktu. Podobne sa má automaticky načítať aj percentuálna zľava podľa typu obchodu. Tieto dve polia musí byť používateľ schopný meniť. V jednej objednávke sa môžu predať viacero produktov jednému klientovi. Okrem týchto informácií sa musí vypísať aj suma jednotlivo za produkt s DPH aj bez, a aj samotné DPH. V prípade, že sa predáva viac produktov, tak vypíše aj celkovú sumu.

Do zoznamu predajov má prístup každý používateľ, kde sa vyskytujú všetky jeho objednávky, ktoré môže filtrovať alebo vyhľadávať podľa reťazca a následne výsledky exportovať v rôznych formátoch. Okrem normálnych informácií, ktoré sa zadávali pri vytváraní objednávky, tak v tomto zozname pri jednotlivej objednávky vypisuje ešte množstvo a cenu spolu. Pri každej takej objednávke sú možnosti upravenia, vymazania a vytvorenie dodacieho listu. Dodací list musí obsahovať: dátum predaja, všetky predané produkty, množstvo, ich predajnú cenu s DPH aj bez, miesto pre pečiatky jak odberateľa, tak dodávateľa, číslo dodacieho listu, číslo objednávky a nakoniec celkovú sumu s DBH a bez. Takýto dodací list musí byť vo formáte .xlsx.

Administrátorská sekcia u predajov rozširuje pôvodnú tabuľku zoznamu predajov. V tabuľke vidí všetky objednávky spolu s tým, kto danú objednávku vytvoril.

2.6 Informačné systémy s podobným zameraním

Systém, ktorý by spĺňal všetky požiadavky uvedené v sekciách 2.1 až 2.5, pravdepodobne ešte neexistuje, avšak existujú mnoho takých systémov, ktoré majú podobné zameranie a práve tie budu spomenuté v podkapitolách 2.6.1 až 2.6.4.

2.6.1 ABC Inventory

ABC Inventory[1] umožňuje evidenciu skladu a predajov. Odporúča sa najmä pre menšie podniky. Neumožňuje ale evidenciu výrobu ani správu používateľov, keďže je to systém s prístupom jedného človeka súčasne. Je to taktiež offline systém, takže dáta nie sú synchronizované.

2.6.2 SalesBinder

SalesBinder[2] je online aplikácia, takže je možné k nemu pristupovať z hocijakého počítača s internetom. Poskytuje evidenciu skladu, predaja a dokonca umožňuje registrovať ďalších používateľov, klientov a dodávateľov. Taktiež umožňuje export dát a štatistík v rôznych formátoch, ale rovnako ako ABC Inventory, neumožňuje evidenciu výroby. Je dôležité spomenúť, že poskytuje veľa rôznych funkcií, ktoré väčšina systémov neposkytuje, ako:

- integráciu rôznych API[3] do svojho účtu
- sken čiarového kódu
- sledovanie pohybu predajov/zásielok

2.6.3 StockPile

StockPile[4] je online informačný systém zaoberajúci sa evidenciou skladu a predajov. Pri skladovej evidencii poskytuje pridať sklady, pridať dodávateľov, vytvárať produkty do skladov. Okrem toho poskytuje ešte podrobné vyhľadávanie produktov medzi vytvorenými skladmi. Evidencia predajov má dosť nedostatkom, keďže poskytuje len odobratie počet kusov z daného skladu. Neummožňuje pridávanie klientov, integráciu API ani sken čiarového kódu. Takýto systém sa hodí najmä pre menšie spoločnosti.

2.6.4 Zoho Inventory

Zoho Inventory[5] je jeden z najznámejších online systémov pre evidenciu skladu a predaja. Zameriava sa najmä na skladovú evidenciu. Okrem toho, že poskytuje to čo SalesBinder a ABC Invetory, tak ešte umožňuje nastaviť upozornenia na rôzne situácie, napríklad na produkty, ktoré dochádzajú na sklade alebo na doručené zásielky. Produkty, ktoré boli odoslané sa dajú sledovať, a to aj pomocou mobilu. Rovnako ako aj SalesBinder a ABC Inventory, tak aj tento systém nemá evidenciu výroby, čo bola jedna z hlavných podmienok zadávateľa.

[1] ABC Inventory - http://www.almyta.com
[2] SalesBinder - https://www.salesbinder.com
[3] API - Application programming interface
[4] StockPile - http://www.thecanvus.com
[5] Zoho Inventory - https://www.zoho.com/inventory

Kapitola 3

Rozbor súčasných a použítých technológií

Obsahom tejto kapitole sú stručne popísané použité technológie, ktoré boli použité pri implentácií informačného systému. Najprv sú v sekcii 3.1 popísané technológie použíté na front-ende a v sekcii 3.2 na back-ende. Ďalej je v sekcii 3.3 popísaný jazyk UML spolu s diagramom použitia prípadov. Nakoniec je stručne popísaný ER diagram v sekcii 3.4.

3.1 Front-end

Front-end, taktiež známy ako vývoj na strane klienta, je v skratke produkovanie v jazykoch HTML, CSS a JavaScripte pre webovú stránku, tak aby používatelia s ňou mohli pracovať. S dobou sa každým dňom vylepšujú technológie používajúce k tvorbe front-endu a kvôli tomu musí vývojár ostať vždy v obraze.

Hlavným zameraním front-endu je aby bol pre používateľa ľahko orientovaný, dostupný, relevantný a taktiež responzívny[1]. Responzivita u front-endu je veľmi dôležitý aspekt, keďže používatelia môžu navštíviť webovú stránku z hocijakého zariadenia s rôznym rozlíšením.[7]

3.1.1 HTML

Bol to práve Tim Berners-Lee, ktorý si pri budovaní webu kládol otázku, ako by dáta, ktoré nám webový server pošle mali vyzerať. Napadla mu myšlienka, že by malo ísť predovšetkým o informácie, ktoré budú hlavne popisovať vzhľad a práve vtedy sa rozhodol vzkriesiť HTML. HTML je skratkou pre Hypertext Markup Language, je založený na značkách, ktorá predstavuje jeden príkaz a zapisujú sa do ostrých zátvoriek.

```
<p>Odstavec</p>
```

Vo vyššie uvedenom príklade sa jedná o značku <p>, ktorá vytvára nový odstavec. Text uvedený medzi značkou je text, ktorý sa objaví na webe. Značky môžu ale obsahovať parametry.

```
<p align="left">Odstavec</p>
```

Tieto parametre popisujú poprípade upresňujú význam daných značiek. Každý parameter má vlastnú hodnotu, ktorá sa prideľuje pomocou znamienka rovnosti.[4]

[1] Responzivita je front-end webovej stránky, ktorý sa prispôsobí veľkosti obrazovky

3.1.2 CSS

CSS je skratkou pre Cascading Style Sheets, ide o jednoduchý dizajnový jazyk, ktorého úlohou je zjednodušiť proces prezentovania webových stránok. CSS má na starosti výzor webovej stránky. Umožňuje meniť farbu textu, pozadia, štýl písma, veľkosť paragrafov a mnoho ďalších vizuálnych efektov. CSS je vytvorené a spravované skupinou ľudí známa, ako CSS Working Group v spoločnosti W3C[2]. Štýlové pravidlá CSS sú interpretované webovým prehliadačom a potom sú aplikované na jednotlivé prvky. Pravidlo štýlu sa skladá z troch častí:

```
Selektor { Vlastnost: Hodnota; }
```

- `Selektor` – Selektor je ľubovoľná HTML značka, na ktorú bude aplikovaný štýl.
- `Vlastnosť` – Vlastnosť je typ parametru HTML značky.
- `Hodnota` – Hodnota sa priraďuje danému parametru.

Príklad funkčného pravidlá vyzerá takto:

```
p { color: #000000; }
```

Vyššie uvedené pravidlo definuje čiernu farbu písma značky <p>. CSS poskytuje hneď niekoľko výhod pri webových stránkach, jedná z tých známejších je, že sa stránky načítavajú rýchlejšie oproti vkladaniu vlastností do HTML značiek. Ďalšou z výhod je ľahká údržba, kompatibilita na rôznych zariadeniach a taktiež ušetrí čas. [17]

3.1.3 Less

Less (Leaner Style Sheets) bol vytvorený v roku 2009 Alexisom Sellierom, ako open-source. Prvá verzia bola napísaná v Ruby, v ďalších verziách sa prešlo na JavaScript. Less je dynamický preprocessor, ktorý rozširuje možnosti CSS. Poskytuje funkcionality, ako premenné, funkcie, operácie, ktoré nám umožnia vytvoriť dynamické CSS. [18] Jedna z hlavných výhod je, že Less umožňuje prehliadaču kompliáciu v realnom čase. Ako už bolo vyššie spomenuté, Less umožňuje definovanie premien. Premenné sa definujú pomocou zavináča. [13]

```
@green-color { color: #2ecc71; }
```

Následná aplikácia daného pravidla môže vyzerať takto.

```
p { color: @green-color; }
```

Vďaka tomu, nemusíme prepisovať vo všetkých selektoroch farbu, ale stačí ak ju zmeníme u premennej, čo nám ušetrí čas.

3.1.4 Bootstrap

Bol vyvinutý Markom Ottom a Jacobom Throntonom v roku 2011 ako Twitter Blueprint pre spoločnosť Twitter. Dnes je bootstrap momentálne jeden z najviac populárných front-end frameworkov[3]. Je to intuitívny, elegantný framework, ktorý využíva HTML, CSS a taktiež JavaScript, ktorý je viac popísaný v podkapitole 3.1.5. Bootstrap poskytuje užitočné funkcionality vo webdizajne. Tie najvýznamnejšie sú:

[2] W3C - World Wide Web Consortium
[3] framework - softwarová štruktúra, ktorá slúži, ako podpobra pri programovaní

- `Responzivita` – Znamená, že k webu je možné pristupovať z hocijakého zariadenia bez ohľadu na velkosť rozlíšenia a web bude prispôsobený danému rozlíšeniu.
- `Podpora prehliadačov` – Bootstrap je podporovaný všetkými prehliadačmi.
- `Flat dizajn` – Dvojrozmerný, jednoduchý, prehľadný a ľahko čitateľný dizajn, ktorý sa ľahko aplikuje pomocou bootstrapu.
- `Grid` – Bootstrap používa dvanásť-stĺpcový systém, ktorý funguje na všetkých zariadeniach, kde sa automatický sám prispôsobuje, keďže po zmene rozlíšenia sa stĺpce preskupia.
- `jQuery pluginy` – Balíček bootstrap obsahuje taktiež niekoľko veľmi praktických pluginov.

Tento front-end framework obsahuje sadu vopred pripravených štýlov, ktoré sa ľahko aplikujú na HTML značky a vzniká, tak responzívny a väčšinou aj moderný web. [16]

3.1.5 Javascript

Po boku HTML a CSS je JavaScript dnes neoddeliteľnou súčasťou všetkých webových prehliadačov. Javascript je multiplatformový, objektovo orientovaný, skriptovací jazyk, ktorý bol štandardizovaný asociáciou ECMA[4]. Jeho syntax vychádza z jazyka C, niektoré jeho rozhrania sa podobajú Jave, funkcionálny a objektový model je inšpirovaný jazykmi Scheme[5] a Self[6]. [22] Programy napísané v JavaScripte sú nezávislé na platforme, to znamená, že ak ich bude podporovať prehliadač, ktorý ich interpretuje, tak sa spustia. JavaScript umožňuje vytvárať dynamické stránky, ktoré menia svoju podobu na základe rôznych udalostí, ako napríklad pohyb myškou. [11] Nižšie je ukážka kódu 3.1, kde sa zmení farba textu po namierení kurzoru na HTML značku s identifikátorom priklad.

```
1  <h1 id="priklad">Text</h1>
2
3  <script>
4  document.getElementById("priklad").onmouseover = function() {
5      mouseOver();
6  };
7
8  function mouseOver() {
9    document.getElementById("priklad").style.color = "red";
10 }
11 </script>
```

Zdrojový kód 3.1: Zmena farby po namierení kurzoru

Vďaka JavaScriptu vzniklo mnoho frameworkov, tie najznámejšie sú: React.js[7], AngularJS[8] a Vue.js[9]. Jednou s najznámejšou JavaScript knižnicou na manipuláciu DOMu[10], je práve jQuery, ktorá je viac popísaná v podkapitole 3.1.6

[4] ECMA - http://www.ecma-international.org
[5] Scheme - http://www.scheme-reports.org
[6] Self - http://www.selflanguage.org
[7] React.js - https://reactjs.org
[8] AngularJS - https://angularjs.org
[9] Vue.js - https://vuejs.org
[10] DOM - Document Object Model

3.1.6 jQuery

Ako už bolo v kapitole 3.1.5 spomenuté, jQuery je JavaScript knižnica určená k manipulácií HTML, spracovaniu udalostí, CSS animáciam. Jedná sa o open-source projekt vytvorený Johnom Reisgom spolu s tímom JavaScript vývojárov. Jeho syntax je ľahko naučiteľný, dnes už je podporovaný všetkými prehliadačmi a existuje veľa pluginov, ktoré rozširujú funkcionalitu jQuery. [2] Jeden z najznámejších pluginov je DataTables[11], ktorý poskytuje responzívne tabuľky s možnosťami ako vyhľadávať, filtrovať, zoradiť dáta v rámci danej tabuľky alebo exportovať dáta v rôznych formátoch ako .csv, .xlsx a .pdf.

Okrem iného jQuery ponúka použitie technológie známej ako AJAX, ktorá značne uľahčuje dynamickú zmenu obsahu webovej stránky, viac o tejto technológií je popísané v podkapitole 3.2.2.

3.2 Back-end

Vývoj na strane servera (back-end) je vlastne to, ako stránka funguje, aktualizuje, pridáva a maže dáta, ktoré sú väčšinou uložené v nejakej databáze. Spracované dáta posielajú na front-end, kde sa zobrazujú. Back-end je zložený zo servera, databázy a softvéra na prepojenie front-endu a back-endu. Server poskytuje priestor pre súbory, zabezpečenie, šifrovanie a služby, ako napríklad email. Databáza je takzvaný rozum, ktorý robí stránku dynamickú. Vždy ak sa odošle požiadavka, či už to je nejaké vyhľadávanie alebo vkladanie dát, je to práve databáza, ktorá ju príjme, vyberie dáta a odošle ich späť stránke. Jedne z najpopulárnejších databáz sú MySQL[12], ktorá je viac popísaná v podkapitole 3.2.3 alebo MongoDB[13]. [20]

3.2.1 PHP

PHP je hypertextový preprocesor na strane servera, ktorý vo väčšine prípadov slúži na vytváranie webových aplikácií spolu s webovým serverom, ako je napríklad Apache[14]. Môže byť použitý aj pre skripty v príkazovom riadku, tie sa ale vyskytujú menej. Kód PHP sa môže vkladať priamo do hypertextových stránok, to môže aj JavaScript, ale narozdiel od JavaScriptu, PHP skript sa interpretuje na strane servera. [3]

Striktne povedané, PHP sa nezaoberá s manipuláciou DOMu alebo udalosťami, respektíve sa nezaoberá ani s tým, ako stránka vyzerá. Poväčšine to čo robí PHP je pre používateľa neviditeľné, pretože výsledok skriptu je zvyčajne HTML kód.

PHP bolo inšpirované najmä jazykmi, ktoré podporovali procedurálne programovanie. Jeho syntax je podobný jazykom C a Perl[15].

```
1 <?php
2 echo "Hello World!";
```

Zdrojový kód 3.2: Ukážka PHP kódu

Okrem iného PHP podporuje veľké množstvo protokolov ako je POP3, IMAP a LDAP, pre ktoré už má vopred definované funkcie. [15]

[11] DataTables - https://datatables.net
[12] MySQL - https://www.mysql.com
[13] MongoDB - https://www.mongodb.com
[14] Apache - https://www.apache.org
[15] Perl - https://www.perl.org

3.2.2 AJAX

Ak chce webová aplikácia komunikovať so serverom, musí kvôli tomu zobraziť respektíve obnoviť stránku. Vo väčšine prípadov to môže byť nepraktické, napríklad pri hlasovaní v ankete. Technológia AJAX je skratkou pre Asynchronous JavaScript and XML, ktorá nám umožňuje komunikovať so serverom asynchrónne bez obnovenia stránky za pomoci JavaScriptu.

Asynchrónna komunikácia znamená, že prehliadač nečaká na vybavenie požiadavku a pokračuje v normálnej práci. Počas tejto doby môže vytvoriť ďalšie požiadavky, čo môže viesť, že server ich vráti v inom poradí než v akom ich dostal.

Požiadavky od AJAXu komunikujú so serverom pomocou protokolu HTTP[16], čo znamená, že môžu klásť požiadavky typu `GET` a `POST`. `GET` sa použije pre získanie dát, `POST` sa použije pre vykonávania operácií, napríklad ukladanie dát. Okrem týchto dvoch metód `HTTP` definuje ešte niekoľko, ako napríklad `PUT` alebo `HEAD`, ktorých spôsob spracovania požiadavkov je obdobný. Ak sa vytvára požiadavok pomocou JavaScriptom, tak môžeme definovať v akom formáte sa budú odosielať dáta na server. Okrem bežného formátu sa môže použiť napríklad formát XML[17] alebo JSON[18]. XML sa vo väčšine prípadov hodí pre dáta so zložitou štruktúrou, naopak JSON sa hodí pre obecné dáta. AJAX požiadavky sa dajú identifikovať podľa hlavičky X-Requested-With s hodnotou XMLHttpRequest, čo sa môže využiť pre kontrolu v PHP skripte či naozaj ide o AJAX požiadavku. [19]

```
1 <?php
2 if ($_SERVER["HTTP_X_REQUESTED_WITH"] == "XMLHttpRequest") {
3     // odoslanie dat
4 }
```

Zdrojový kód 3.3: Identifikácia AJAX požiadavky

3.2.3 MySQL

MySQL bola pôvodne vyvinutá, aby mohla ovládať veľké databázy, väčšou rýchlosťou, než v tej dobe existujúce riešenia. MySQL je relačný databázový systém, čo vlastne znamená, že sa jednotlivé údaje môžu ukladať na rôzne miesta, ktoré je možné potom prepojiť.

Samotné informácie sa ukladajú do tabuliek a v nerelačných databázových systémov sa ukladajú do jednej veľkej oblasti. Každá tabuľka sa skladá zo stĺpcov, ktoré zastupujú jednotlivé časti uloženej informácie. V stĺpci sú uložené údaje, ako mená používateľov, heslá, rodné čísla a iné. Každý taký stĺpec môže mať rôzny typ informácií, napríklad text, čísla alebo dátum. [9]

Je možné nastaviť vzťahy medzi tabuľkami k viazaniu dát, ktoré spolu súvisia a sú umiestnené v rôznych tabuľkách. Každý taký vzťah je definovaný stupňom, kardinalitou a voliteľnosťou. Databáza uplatňuje tieto pravidlá, respektíve vzťahy, takže pri dobrej navrhnutej databáze nikdy nedôjde k duplikácii, nekonzistencii alebo neaktuálnosti dát. [8]

S verziou 5.0 prišli MySQL pohľady, ktoré môžu zlepšiť výkon vďaka tomu, že pohľad bude vracať len tie stĺpce, ktoré naozaj programátor potrebuje. Táto databázová štruktúra je vlastne virtuálna tabuľka, ktorá neobsahuje žiadne dáta. Pohľady rozdeľujeme na dve typy podľa toho z koľkých tabuliek vznikli a akým spôsobom, na jednoduché a komplexné. Jednoduchý pohľad je vytvorený z jednej tabuľky a zase naopak komplexné z niekoľkých,

[16] HTTP - Hypertext Transfer Protocol
[17] XML - eXtensible Markup Language
[18] JSON - JavaScript Object Notation

a taktiež u komplexných pohľadoch môžu obsahovať aj funkcie. Vo funkčnej databáze môže nastať taký prípad, kde používatelia posielajú veľa príkazov typu SELECT, ktorého výsledkom je sekvenčné čítanie veľkých tabuliek čo výrazne pohorší výkonu. Na to sa môže použiť pohľad s klauzulou WHERE, ktorý obmedzí počet vrátených riadkov, vďaka čomu sa zlepší výkon. [12]

```
1 CREATE VIEW sales_view AS
2 SELECT order_id, SUM(quantity * price) total
3 FROM orders
4 GROUP by order_id
5 ORDER BY total DESC;
```

Zdrojový kód 3.4: Ukážka vytvorenia jednoduchého MySQL pohľadu

Existujú minimálne 2 možnosti, ako pracovať s databázou MySQL. Jedna možnosť je pomocou zadávaním príkazov priamo u servera. Na to je ale potrebné mať dobré znalosti SQL. Druhá možnosť je použitie phpMyAdmin[19], ide o bezplatný PHP program, ktorý beží na webovom prehliadači. Nevyžaduje až tak veľké znalosti ako pri prvej možnosti, šetrí čas a uľahčuje prácu. Okrem toho umožňuje exportovať dáta v rôznych formátoch ako .csv, .pdf alebo .sql, importovať dáta vo formátoch .csv alebo .sql. a taktiež vytváranie a spravovanie používateľov. [6]

3.3 UML

UML je univerzálny jazyk pre vizuálne modelovanie systémov, ktorý bol navrhnutý aby spojil najlepšie postupy modelovacích techník a softwarového inžinierstva. Okrem toho je explicitne navrhnutý takým spôsobom, aby ho mohli implementovať všetky nástroje CASE.[20].

Je nezávislý na hocijakom programovacom jazyku a platforme. Pre čisto objektovo orientované jazyky, ako sú Java, C Sharp alebo Smalltalk poskytuje znamenitú podporu. Avšak poskytuje podporu aj keď menšiu pre hybridné objektovo orientované jazyky, ako je napríklad C++.[1]

UML sa skladá z grafických prvkov, ktoré sa podľa pravidiel jazyka dajú vzájomne kombinovať do podoby 13 diagramov, ktoré sú:

- **Diagramy chovania** - Behavior diagrams
  - **Diagram prípadov použitia** - Use case diagram
  - **Stavový diagram** - State chart (state machine) diagram
  - **Diagram činností** - Activity diagram
- **Štrukturálne diagramy** - Structure diagrams
  - **Diagram tried** - Class diagram
  - **Diagram objektov** - Object diagram
  - **Diagram komponentov** - Component diagram
  - **Kombinovaný štrukturálný diagram** - Composite structure diagram
  - **Diagram balíčkov** - Package diagram

[19] phpMyAdmin - https://www.phpmyadmin.net
[20] CASE - Computer-Aided Software Engineering

  - **Diagram nasadenia** - Deployment diagram

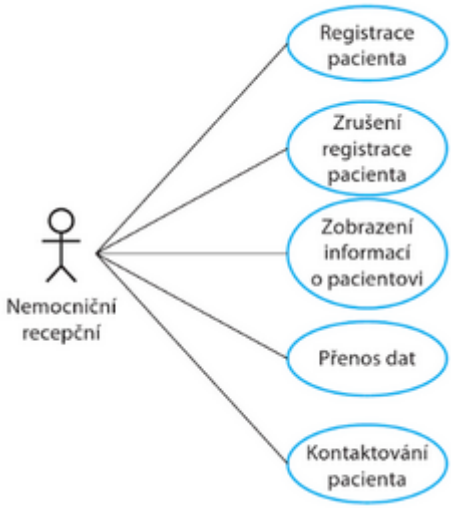
- **Interakčné diagramy** - Interaction diagrams
  - **Diagram spolupráce** - Collaboration diagram
  - **Diagram komunikácií** - Communication diagram
  - **Sekvenčný diagram** - Sequence diagram
  - **Prehľadový diagram interakcií** - Interaction overview

Z vyššie uvedených diagramov je najčastejšie používaný práve diagram prípadov použitia (use-case diagram) [10].

3.3.1 Use-case diagram

Use-case diagram alebo diagram prípadov použitia sa používa na podporu zisťovania požiadavkov. Popisuje väzby medzi aktérmi a prípadmi použitia systému. Súbor prípadov použitia vymedzuje hranice systému špecifikáciou takzvanej „funkčnej obálky“ systému, pomocou ktorej server komunikuje so svojím okolím. [21]

Je to vlastne nejaký scenár, ktorý popisuje čo používateľ očakáva od systému. Každý prípad použitia reprezentuje diskrétnu úlohu, ktorá zahrňuje externú interakciu so systémom. Prípad použitia má väčšinou formu elipsy. Aktér zvyčajne predstavuje používateľskú rolu alebo zastupuje spolupracujúci systém, takýto aktér sa reprezentuje v dokumentácií vo forme postavičky.



Obr. 3.1: Ukážka prípadu použitia aktéra: nemocničný recepčný

Na obrázku 3.1 je znázornený jednoduchý príklad prípadu použitia jedného aktéra, ktorý môže vykonávať niekoľko úloh v systéme, ako: registrácia pacienta, zrušenie registrácie pacienta, zobrazenie informácií o pacientovi, prenos dát a kontaktovanie pacienta. [14]

3.4 Entitne-relačný model

Entitne-relačný model (ER model) je definovaný ako množina pojmov, s ktorými popisujeme príslušný systém za účelom špecifikácie presnej štruktúre databázy. Je zložený z entít, vzťahov a atribútov. Predtým ako bude vysvetlený proces návrhu systému je nutné definovať tieto pojmy.

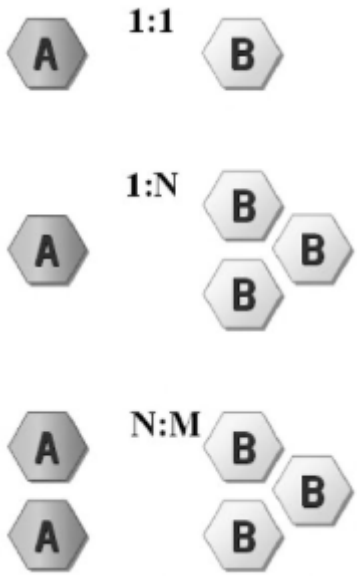
- **Entita** - je objekt reálneho sveta, ktorý môže existovať nezávislé a je jasne rozdielny od ostatných objektov.
- **Vzťah** - je väzba medzi dvoma alebo viacerými entitami.
- **Atribút** - je funkcia, ktorá priraďuje hodnotu jednotlivým entitám alebo vzťahom. Zvyčajne atribút určuje niektorú podstatnú vlastnosť.

Proces návrhu systému spočíva v niekoľkých jednoduchých krokov:

1. Identifikácia typov entít ako množina objektov rovnakého typu.
2. Identifikácia typov vzťahov, do ktorých dané entity budú vstupovať.
3. Priradenie atribútov, ktoré bližšie popisujú vlastnosti daných entít a vzťahov.

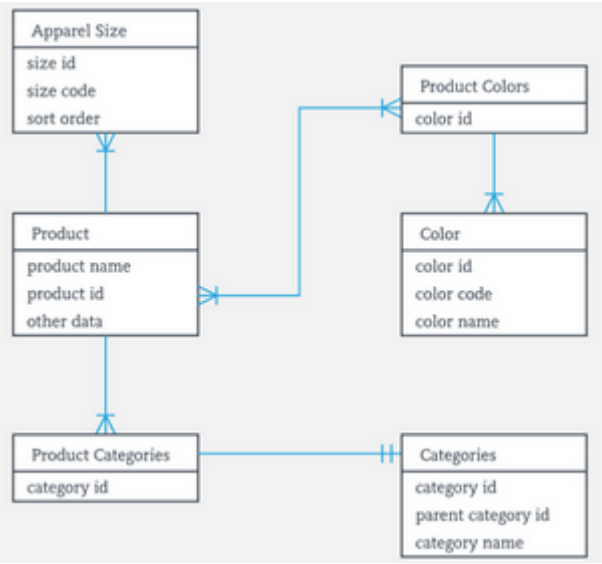
Pri návrhu databázových tabuliek, je možné ešte definovať kardinalitu vzťahov medzi entitami. Druhy vzťahov môžu byť:

- **1:1** - prvej entite odpovedá maximálne jedna entita.
- **1:N** - prvej entite odpovedá viac entít, naopak druhej entite odpovedá maximálne jedna entita.
- **M:N** - prvej entite odpovedá viac entít, rovnako to má aj druhá entita.



Obr. 3.2: Ukážka kardinality

Kardinalita slúži najmä na zistenie či vzťah jednej entity k výskytu inej entity je výnimočný a môže nastať iba raz alebo viackrát. [5]



Obr. 3.3: Ukážka ER diagramu s kardinalitou

Kapitola 4

Návrh informačného systému

Obsahom tejto kapitoly je návrh informačného systému, ktorý som navrhol z analyzovaných požiadavkách spomenuté v kapitole 2. Súčasťou návrhu sú use-case diagramy popisujúce aktérov systému a schéma databázy. Systém som rozdelil na štyri hlavné časti ako už bolo spomenuté pri analýze, ich návrh je popísaný v sekciách 4.3 až 4.6. Nakoniec popíšem návrh front-endu, uchovávanie a spracovanie dát. Po popísaní týchto návrhov bude možné začať s implementáciou systému.

4.1 Use-case diagram

Podľa analýzy požiadavkov zákazníka čo by mal systém umožňovať používateľom na základe ich práv, som vytvoril tri use-case diagramy, kde každý jeden popisuje jednotlivého aktéra. Čo môže v systéme, ktorý aktér vykonávať bolo nesúvislo spomenuté už v kapitole 2, a preto som sa rozhodol to spísať, tak aby to bolo prehľadné a to s pomocou use-case diagramov. Spomínané use-case diagramy sú:

1. **Use-case diagram aktéra - Predajca** - Predajca je v podstate každý zaregistrovaný používateľ systému. Jeho hlavnou úlohou je vytvárať klientov, ktorým môže následne predať produkty vytvorením objednávky. Konkrétne interakcie so systémom nad jednotlivými kategóriami je možné vidieť na use-case diagrame v prílohe B.1.

2. **Use-case diagram aktéra - Správca skladu** - Správca skladu okrem toho, že má rovnaké práva ako predajca, tak má ešte práva k niekoľkým kategóriám, ako je možné vidieť na use-case diagrame v prílohe B.2. Hlavnou úlohou správcu skladu je vytvárať objednávky do skladu zásob, pridávať dodávateľov a produkty. Ďalšie interakcie je možné vidieť na už spomenutom use-case diagrame.

3. **Use-case diagram aktéra - Administrátor** - Úlohou administrátora v tomto informačnom systéme je vlastne to čo je úlohou manažéra v spoločnosti. Jednou z tých úloh je pozorovať čo sa vykonáva v systéme, to môže vďaka prístupu k záznamom. Ďalej môže pridávať používateľov do systému, ktorým môže vytvoriť rôzne úlohy a nakoniec si môže zobraziť štatistiku pre každého používateľa zvlášť. Viac špecificky popísané interakcie pre spomenuté kategórie sú znázornené v use-case diagrame zobrazeného v prílohe B.3.

4.2 Schéma databázy

Neoddeliteľnou časťou každého informačného systému je databáza. Najprv podľa požiadavkov bol vytvorený ER diagram a následne na základe analýzy cieľovej domény bola navrhnutá schéma databázy. Na to aby bol informačný systém dobre implementovaný, je potrebné mať dobre navrhnutú databázu, ktorá k tomu bude viesť. Zle navrhnutá databáza môže viesť k problémom jak pri implementácií, tak vo funkčnosti už implementovaného systému.

V ďalších sekciách sú bližšie popísané štyri hlavné časti systému. Konkrétne, ktoré tabuľky sú potrebné k funkčnosti danej časti, rovnako ako na čo slúžia jednotlivé stĺpce. Spomenuté tabuľky a ich stĺpce vychádzajú zo schémy databázy, ktorú som navrhol a je možné ju vidieť v prílohe B.4.

4.3 Používatelia systému

Návrh tejto časti poskytne prehľad rolí v informačnom systéme, aké data sa ukladajú týkajúce sa používateľov do databázy. Na to aby táto časť bola funkčná je potrebné vytvoriť päť tabuliek, ktoré sú:

1. **User** - Táto tabuľka obsahuje používateľov informačného systému. Obsahuje data, ako prihlasovacie meno, heslo, email, dátum registrácie, celé meno, telefónne číslo a počet predajov, ktoré vykonal. Ďalej obsahuje aj niekoľko nezvyčajných stĺpcov, ktoré je dobre spomenúť. Jeden z nich je `salt`, ktorý sa používa na vytvorenie zašifrovaného hesla. Ďalši stĺpec je `ugroup`, ktorý obsahuje práva používateľa.

2. **User_session** - Táto tabuľka slúži na zapamätanie si prihlásenia používateľa na určitú dobu. Obsahuje stĺpec `user_id` , čo je id používateľa a `hash` je vytvorený hash pri prihlásení.

3. **Groups** - Sem sa ukladajú jednotlivé role informačného systému. Obsahuje meno role a stĺpec menom `permissions`, ktorý obsahuje práva vo formáte JSON.

4. **Task** - Tabuľka `Task` obsahuje jednotlivé úlohy pre používateľov systému. Je tvorená stĺpcami: `user` - slúži na uloženie id používateľa, ktorému bola úloha pridelená, `date` a `finish_date` - tieto stĺpce slúžia na uloženie dátumu vytvorenia a deadlinu úlohy, `status` - hovorí v akom stave je daná úloha, napríklad splnená alebo nesplnená, stĺpec `priority` - definuje prioritu pridelenej úlohy, napríklad mála, stredná alebo veľká a nakoniec stĺpec `note`, ktorý obsahuje podrobný popis úlohy.

5. **Log** - Ako vyplýva z názvu tabuľky, táto tabuľka obsahuje záznamy činností, ktoré sa vykonajú v systéme. Hlavné stĺpce tejto tabuľky sú `user` - čo je id používateľa, ktorý vykonal nejakú činnosť v systéme, `date` - dátum vykonania, `category` - obsahuje textovú informáciu, v ktorej kategórií bola vykonaná činnosť a `message`, ktorý obsahuje popis vykonanej činnosti.

V tejto častí ešte existuje tabuľka **information**, ktorá je spojená s vyššie spomenutými tabuľkami ale nie je závislá na funkčnosti tejto časti. Tabuľka **information** obsahuje stĺpce na uloženie informácií o spoločnosti a ešte obsahuje jeden dôležitý údaj, ktorý sa využíva pri výpočtoch, ide o DPH.

4.4 Evidencia skladu

Táto časť poskytuje aké údaje sa ukladajú do databázy potrebné na kompletnú evidenciu v rámci skladu. Pre úplnú funkčnosť tejto časti sa počíta s funkčnosťou prvou časťou tj. Používatelia systému. Tvoria ju šesť tabuliek, ktoré sú:

1. **Supplier** - V tejto tabuľke sa nachádzajú dodávatelia, ktorí sa využívajú na to aby sa pri evidencii objednávkach vedelo od koho je zásoba. V stĺpoch uchováva dáta o dodávateľoch, ako meno, adresa, informácie o firme, kontaktné údaje v stĺpoch začínajúce sa na `contact_` a poznámku o dodávateľovi v stĺpci `note`. V stĺpci `date` sa eviduje dátum pridania.

2. **Category** - Táto tabuľka je jedna z troch tabuliek, ktoré slúžia na členenie zásob. Konkrétne v tejto tabuľke sa uchováva v stĺpci `name` meno kategórie a metrická jednotka v stĺpci `unit`.

3. **Subcategory** - Rovnako ako tabuľka **Category**, aj táto slúži na členenie zásob. Obsahuje cudzí kľúč odkazujúci do tabuľky **Category** v stĺpci `category`, ktorý hovorí o tom do ktorej kategórie patrí podkategória. Okrem toho obsahuje ešte dve stĺpce `name` a `date`, ktoré predstavujú meno a dátum pridania.

4. **Item** - Táto tabuľka popisuje konkrétnu položku a rovnako ako tabuľka **Subcategory** obsahuje cudzí kľúč, ale tento odkazuje do tabuľky **Subcategory**, nachádzajúci sa v stĺpci `subcategory`. Ďalej obsahuje aj meno položky v stĺpci `name` a dátum pridania `date`. Na rozdiel od tabuliek **Category** a **Subcategory**, táto tabuľka ešte obsahuje stĺpec menom `amount`, ktorý uchováva množstvo položky.

5. **Storage_supply_order** - Obsahom tejto tabuľky sú jednotlivé objednávky do skladu zásob. V stĺpcoch `date` a `date_approved` sa uchováva dátum vytvorenia objednávky a dátum v aký deň bola objednávka potvrdená. Okrem dátumu potvrdenia sa v stĺpci `approved` ukladá informácia či je objednávka potvrdená alebo nie. Ďalej obsahuje stĺpec `lock_in`, ktorý zaistí aby nebolo možné vymazať danú objednávku, ak z nej boli použité zásoby pri výrobe produktu. Obsahuje aj cudzí kľúč, ktorý odkazuje na tabuľku **user** v stĺpci `employee`, ktorý uchováva informáciu o tom kto vytvoril objednávku. Súčasťou tejto tabuľky je aj suma všetkých zásob v objednávke v stĺpci `summary`.

6. **Storage_supplies** - Keďže v jednej objednávke sa môže objednať viac zásob, tak táto tabuľka musí existovať. Skladá sa z piatich cudzích kľúčov, prvé tri slúžia na členenie objednanej zásoby, sú to `category`, `subcategory` a `item`, ďalši cudzí kľúč `supplier` odkazuje na tabuľku **Supplier**, ktorý slúži na uchovanie od koho bola zásoba objednaná. Podstatou posledného cudzieho kľúča `storage_supply_order_id` je aby sa vedelo, ku ktorej objednávke zásoba patrí. Okrem cudzích kľúčov obsahuje ešte štyri stĺpce, `amount_save` - pre objednané množstvo zásoby, `price` - uchováva cenu za jednotku zásoby, `sum` - suma zásoby a nakoniec `amount` - pre uchovanie aktuálneho množstva, ktorý sa využíva pri výrobe produktu.

Súčasťou tejto časti je aj pohľad **storage_supply**, ktorý predstavuje sklad zásob. Je vytvorený stĺpcami z troch tabuliek: **Category**, **Subcategory** a **Item**.

4.5 Evidencia výroby

Tretia časť systému je evidencia výroby, ktorá uchováva všetky informácie týkajúce sa výroby. Podobne ako pri evidencii skladu, na to aby táto časť bola úplne funkčná sa počíta s funkčnosťou minulých dvoch častí. Túto časť tvorí šesť tabuliek:

1. **Product_category** - Táto tabuľka slúži na roztriedenie produktov s rôznym dátumom spotreby do kategórií. Obsahuje meno kategórie v stĺpci `name` a dátum vytvorenia v stĺpci `date`.

2. **Manufacture_product** - Každý produkt, ktorý bol vytvorený používateľom sa uchováva v tejto tabuľke. Obsahuje jeden cudzí kľúč - `category_id`, ktorý odkazuje do tabuľky **Product_category**, čo vlastne hovorí do ktorej kategórie patrí vyrobený produkt. Ďalej tabuľka uchováva dátum pridania a spotreby v stĺpcoch `date` a `expiration_date`, trvanie výroby v stĺpci `duration`, šaržu v stĺpci `batch`, predajnú cenu v stĺpci `sale_price` a vyrobené množstvo v stĺpci `amount`. Podobne ako tabuľka **Storage_supplies**, tiež obsahuje `amount_save`, ktorá uchováva aktuálne množstvo produktu.

3. **Manufacture_product_employee** - Ako vyplýva z názvu tabuľky, jedná sa o zamestnancov, ktorí sa podieľali na výrobe produktu. Obsahuje jeden cudzí kľúč - `manufacture_product_id`, ktorý odkazuje na tabuľku `Manufacture_product_id`, slúži na to aby sa vedelo, ku ktorej objednávke zamestnanec patrí. Okrem toho obsahuje ešte stĺpce, ktoré uchovávajú meno zamestnanca, počet odrobených hodín a mzda v stĺpoch `name`, `hour` a `wage`.

4. **Manufacture_product_energy** - Podobne ako tabuľka pre zamestnancov, aj táto slúži na evidovanie spotrebovaných energií pri výrobe produktu. Obsahuje ten istý cudzí kľúč - `manufacture_product_id` ako tabuľka pre zamestnancov. Ďalej obsahuje stĺpec `type`, ktorý môže nadobúdať tri hodnoty, podľa ktorých sa zistí o akú energiu sa jedná. Stĺpce `amount` a `price` evidujú použité množstvo a cenu za jednotku spotrebovanej energie.

5. **Manufacture_product_material** - Táto tabuľka popisuje využité zásoby zo skladu zásob, ktoré boli použité na výrobu produktu. Obsahuje dve cudzie kľúče, jeden z nich je `manufacture_product_id`, ktorý má rovnakú funkciu ako pri zamestnancov a energii. Druhý je `storage_supplies_id`, ktorý odkazuje na tabuľku `storage_supplies`, slúži na to aby sa vedelo aká zásoba sa použila. Okrem toho sa ešte eviduje použité množstvo a cena za jednotku v stĺpcoch `amount` a `price`.

6. **Manufacture_product_other** - Posledná tabuľka v evidencii výroby slúži na výdaje pri výrobe, ktoré nie su očakávane, napríklad výpadok elektriny. Rovnako ako minulé tabuľky, aj táto obsahuje cudzí kľúč `manufacture_product_id` s rovnakou funkcionalitou. Okrem toho má ešte dva stĺpce, jeden na popis výdaju - `info` a druhý na cenu výdaju - `price`.

Rovnako ako je pohľad pri evidencii skladu, tak aj tu je pohľad **Storage_product**, ktorý slúži na prehľad aktuálnych produktov na sklade. Skladá sa z dvoch tabuliek **Manufacture_product** a **Product_category**.

4.6 Evidencia predajov

Posledná časť je venovaná všetkému čo sa týka evidencii predajov. Táto časť počíta už s funkčnosťou všetkých ostatných častí. Je zložená z šiestich tabuliek:

1. **Store** - Táto tabuľka slúži na uchovávanie informácií o obchodoch, do ktorých patria jednotliví klienti. Obsahuje dátum pridania v stĺpci - `date` a názov obchodu `name`.

2. **Store_discount** - Obsahom tejto tabuľky sú zľavy pre jednotlivé kategórie produktu. Má tri stĺpce a z toho dve sú cudzie kľúče, prvý je `store_id`, ktorý odkazuje na tabuľku **Store**. Druhý cudzí kľúč je `product_category_id` tento odkazuje na tabuľku **product_category**. Nakoniec tretí stĺpec - `discount`, ktorý hovorí o tom aká zľava je nastavená pre jednotlivú kategóriu produktu v danom obchode.

3. **Client** - V tejto tabuľke sa nachádzajú všetci klienti, ktorých pridali používatelia informačného systému. Ako už bolo spomenuté pri špecifikácií požiadavok, klient môže byť buď firemný klient alebo súkromná osoba. Táto informácia sa uchováva v stĺpci `position`. Pokiaľ je klient firemný, tak údaje týkajúce sa firmy sa ukladajú do tabuliek začínajúce sa na `company_`, ide o meno, adresu, ičo, dič, ič-dph a kontakt. Pokiaľ je ale klient súkromná osoba, tak sa uchováva len jeho meno v tabuľke `personal_name`. Pri oboch typoch klienta, sa ešte ukladá ich email, telefónne číslo a dátum pridania v stĺpcoch - `email`, `phone` a `date`. Okrem toho ešte má tri cudzie kľúče, `user_id` - odkazuje na tabuľku **User**, ktorý slúži na to aby sa vedelo kto vytvoril klienta, resp. kto ho spravuje. Druhý cudzí kľúč `store_id` odkazuje do tabuľky **Store**, čím špecifikuje do ktorého obchodu je klient priradený. Posledný, tretí cudzí kľúč je `status_id` odkazujúci do tabuľky `client_status`, ktorý priraďuje status klientovi. V tejto tabuľke je ešte jeden stĺpec - `orders`, ktorý uchováva informáciu o počte objednávok.

4. **Client_status** - Obsahom tejto tabuľky sú statusy vytvorené používateľmi informačného systému. Obsahuje jeden cudzí kľúč `created_by`, ktorý odkazuje na tabuľku **User**, čím špecifikuje tvorcu statusu. Ďalej obsahuje stĺpec `rgb`, ktorý slúži na definovanie farby statusu vo formáte rgb() v jazyku CSS. Okrem farby statusu, sa uchováva meno a dátum pridania statusu v stĺpcoch `name` a `date`.

5. **Sale** - Jedna z najvýznamnejších tabuliek z nielen tejto časti, ale z celého systému je práve táto tabuľka. Ukladá všetky objednávky, ktoré vykonali používatelia. Obsahuje dve cudzie kľúče. Prvý je `user` odkazujúci na tabuľku **User**, čím špecifikuje tvorcu objednávky. Druhý cudzí kľúč `client` odkazuje na tabuľku `Client`, ktorý definuje ktorému klientovi bola vytvorená objednávka.

6. **Sale_products** - Posledná závislá tabuľka v tejto časti slúži na definíciu jednotlivých produktov, ktoré sú obsiahnuté v objednávke. Obsahuje tri dôležité cudzie kľúče. Prvý z nich je `sale_id`, ktorý odkazuje na vyššie spomínanú tabuľku **Sale**. Úlohou prvého cudzieho kľúča je špecifikovať, ku ktorej objednávky patrí predaný produkt. Druhý cudzí kľúč je `product_category_id`, ktorý odkazuje na tabuľku **product_category**, určuje do ktorej kategórie produktov patrí produkt. Posledný cudzí kľúč je `product_id` odkazujúci na tabuľku `manufacture_product`, ktorý definuje konkrétny predaný produkt. Okrem týchto cudzích kľúčov sa ešte uklada predané množstvo, predajná cena a prípadne aj zľava v stĺpcoch `amount`, `price` a `discount`.

Pri tejto časti ešte spomeniem jeden pohľad, ktorý je závislý na všetkych častiach, ide o `informations_view`. Tento pohľad obsahuje len jeden riadok, v ktorých sa nachádza celkový počet dodávateľov, klientov, produktov, predajov za posledný rok, mesiac a týždeň. Je zložený z tabuliek **Supplier**, **Client**, **Manufacture_product** a **Sale**.

Návrh tejto časti skompletizuje štruktúru databázy, vďaka čomu získam prehľad aké dáta a v akom dátovom type pri jednotlivých časti uchovávať. Okrem prehľadu získam aj, ktorú tabuľku s ktorou spojiť, čo sa hodí najmä pri samotnej implementácií systému.

4.7 Používateľské rozhranie

Vzhľadom k požiadavkám zadávateľa na návrh používateľského rozhrania boli vybrané technológie na jeho tvorbu. HTML na tvorbu kostry webovej stránky. Na štýl stránky CSS. Používatelia systému môžu pristupovať na web z hocijakého zariadenia. Ak sú priamo pri nejakom klientovi a chcú skontrolovať či majú na sklade konkrétny výrobok, bude najskôr pristupovať z mobilu a aby sa stránka zobrazila správne, musí byť responzívna. Responzivita sa zaistí pomocou frameworku Bootstrap. Pri návrhu pre interaktívnosť webu použila JavaScriptová knižica - jQuery.

Keďže stránka by mala byť moderná a malo sa v nej ľahko orientovať, zvolil som pre návrh toho ako bude stránka rozložená podľa známych redakčných systémov, ako je WordPress[1], Joomla[2] alebo Drupal[3].

Na zobrazenie dát sa používajú najmä tabuľky. Na stránke sa vyskytujú niekoľko desiatok a aj tieto tabuľky musia byť responzívne, umožňovať vyhľadávanie a exportovanie, na to slúži jQuery plugin - DataTables.

4.8 Uchovávanie a spracovanie dát

Všetky dáta, s ktorými používateľ pracuje sa ukladajú do databázy. Pri návrhu som zvolil databázu MySQL, ktorá sa nachádza na strane servera. Vkladanie dát do databázy umožňuje jazyk PHP s jeho funkciami, ktoré má už vopred vbudované.

Okrem vkladania, PHP umožňuje ešte zobrazenie a spracovanie dát. Vždy ak sa odošle nejaká požiadavka, respektíve nejaké dáta na server, tak na to aby sa zobrazili, sa stránka musí obnoviť. Čo nie je najlepšie riešenie, ak má byť stránka moderná a interaktívna. Na to aby sa vyhlo obnove stránky po každej podobnej situácií, som zvolil technológiu AJAX, ktorá to umožňuje. To, že AJAX požiadavku spracoval a upravil obsah stránky nedáva nijak najavo používateľovi. Preto som využil notifikačný jQuery plugin - Lobibox[4], ktorý upozorní používateľa v takýchto prípadoch.

Pri prenose dát medzi serverom a klientom pomocou AJAXu, sú prenášané data serializované serializačným jazykom JSON. Čo mi poskytuje možnosti, ako: kontrolovať výstup, vstup, upozorniť používateľa o zmene a vykonávať požiadavky nad databázou.

[1] WordPress - https://sk.wordpress.org/
[2] Joomla - https://www.joomla.org/
[3] Drupal - https://www.drupal.org/
[4] Lobibox - http://lobianijs.com/site/lobibox

Kapitola 5

Implementácia

Táto kapitola sa zameriava na implementáciu informačného systému, ktorá vychádza z návrhu v kapitole 4. Najprv je popísaná tvorba databáze podľa jej schémy, potom štruktúra, inicializácia systému a následne časti samotného systému. Predtým ako budú popísané jednotlivé časti systému je popísaný front-end. Jak časti systému, tak front-end budú popísané z programátorského hľadiska. Čo znamená, že bude popísané, čo všetko sa využilo a ktoré jazyky sa použili pri ich tvorbe. Nakoniec bude popísaná z rovnakého hľadiska štatistika.

5.1 Databáza

Databáza je kľúčovou časťou, každého informačného systému a preto som ju naimplementoval pri tvorbe systému ako prvú. Samotná databáza bola vytvorená na MySQL serveri, ktorú poskytuje hosting, na ktorom beží doména. Všetky tabuľky boli vytvorené pomocou phpMyAdmina, ktorý to umožňuje cez grafické rozhranie. Pri ich vytváraní sa zadáva meno tabuľky, polia, ich dátový typ a maximálny počet znakov.

Pohľady boli vytvorené taktiež pomocou phpMyAdmina, ale oproti tabuľkám, pohľady boli vytvorené cez príkazový riadok. Konkrétné pohľady budú popísané pri jednotlivých častiach systému a štatistike.

Vytvorené tabuľky boli naplnené dátami, ktoré sa aspoň približujú reálnym. Tieto dáta poslúžia na demonštráciu chodu systému a taktiež sa potom využijú aj pri testovaní.

5.2 Adresárová štruktúra

Pre lepší prehľad sú jednotlivé funkcionality a zdrojové kódy do niekoľkých adresárov. Okrem prehľadu, niektoré funkcie závisia práve na tomto rozdelení, čo bude viac vysvetlené v sekcii 5.3.

- `css` – obsahuje súbory, ktoré upravujú štýl stránky.
- `js` – v tomto priečinku sa nachádzajú najmä súbory obsahujúce jQuery skripty.
- `pages` – tento priečinok obsahuje podpriečinky pre jednotlivé kategórie systému spomenuté pri analýze, konkrétne 2.2 až 2.5. V podpriečinkoch sa nachádzajú zdrojové súbory zobrazujúce data z databázy spojené s danou kategóriou a formuláre pre pridanie a úpravu záznamov.
- `datatables` – obsahuje všetky tabuľky, ktoré sa nachádzajú v systéme.

- `classes` – obsahuje zdrojové súbory pre jednotlivé triedy.
- `core` – v tomto priečinku sa nachádza jediný súbor - init.php, ktorý slúži na inicializáciu.
- `functions` – obsahuje jednotlivé funkcie v php.
- `form` – podobne ako `pages`, aj tento priečinok obsahuje podpriečinky, ale narozdiel od neho. Tieto podpričinky obsahuju zdrojové súbory, ktoré vykonávajú operácie nad databázou.

Hlavné súbory pre jednotlivé kategórie, v ktorých sa inicializuje, sa nachádzajú v zdrojovom adresári.

5.3 Inicializácia

Inicializácia sa vykonáva v súbore `init.php`, ktorý sa volá v aždom hlavnom súbore pre jednotlivé kategórie. Na začiatku sa volá funkcia `session_start()`, ktorý vygeneruje používateľovi náhodný reťazec tzv. session identifikátor. Vytvorený identifikátor sa pošle používateľovi pomocou cookie a na strane servera sa pod tým identifikátorom ukladajú všetky data, ktoré sú do session premenných uložené.

Zdrojový kód 5.1 je zobrazenie funkcie zo súboru `init.php`, ktorá zaregistruje viacero funkcií, ktoré PHP umiestni do zásobníku a zavolá ich keď je deklarovaná nová trieda. Tým pádom sa všetky triedy načítajú, až keď sa použije nedefinovaná trieda. Jediné čo je potrebné je mať rovnaký názov súboru v priečinku classes ako názov triedy.

```
1 <?php
2 spl_autoload_register(function($class) {
3         require_once __DIR__ ."/../classes/" . $class . ".php";
4 });
```

Zdrojový kód 5.1: Funkcia pre automatické nahrávanie tried

Ďalej súbor `init.php` inicializuje funkcie v priečinku functions. Okrem toho obsahuje ešte konfiguračné údaje potrebné k pripojeniu do MySQL databázy.

5.4 Čisté URL

Väčšina funkcií už predpokladá, že URL je upravené a neobsahuje žiadne špeciálne znaky, takému URL sa hovorí Clean URL[1]. Čisté URL je možné vytvoriť pomocou súboru `.htaccess`, ktorého časť je možné vidieť v zdrojovom kóde 5.2.

```
1 RewriteEngine On
2 Options +FollowSymLinks
3 RewriteCond %{REQUEST_FILENAME} !-d
4 RewriteCond %{REQUEST_FILENAME}.php -f
5 RewriteRule ^(.+?)/([^/]+)(/[^/]+)?/?$ $1.php?page=$2 [L,QSA]
6 RewriteCond %{REQUEST_FILENAME} !-d
7 RewriteCond %{REQUEST_FILENAME}.php -f
8 RewriteRule ^(.*?)/?$ $1.php [NC,L]
```

Zdrojový kód 5.2: Časť kódu pre čisté URL zo súboru .htaccess

[1] Clean URL - https://en.wikipedia.org/wiki/Clean_URL

Zobrazený zdrojový kód 5.2 slúži na upravenie URL adresy. Ak požadovaný súbor končí .php a nie je priečinok, tak prepíše adresu. Sú dva prípady ako môže prepísať adresu:

```
1 http://adresa.eu/users.php
2 http://adresa.eu/users?page=new
```

Zmení na:

```
1 http://adresa.eu/users
2 http://adresa.eu/users/new
```

Okrem súboru `.htaccess` je nutné ešte spomenúť funkciu `createRedirect()`, ktorá je úzko spätá s funkcionalitou `.htaccessu` a ďalej ju využívajú časti systému. Funkcia `createRedirect()` je prítomná vo všetkých hlavných zdrojových súboroch. Jej úlohou je načítať zdrojový kód zo súborov z priečinku `pages` pre jednotlivé kategórie na základe URL. Napríklad ak by bola adresa http://adresa.eu/users/new, tak funkcia načíta zdrojový kód zo súboru umiestneného v pages/users/new.php.

5.5 Front-end

Front-end je implementovaný podľa návrhu to znamená, že je rozdelený na tri časti. Prvá a druhá časť nemení svoj obsah, ide o sidebar a header. Sidebar, respektíve bočný panel slúži ako menu, pomocou ktorého sa môže používateľ presúvať na iné stránky v rámci systému. Header - horný panel, slúží na zobrazenie úloh, upravenia profilu a odhlásenia. Posledná časť zobrazuje dáta z databázy, takže jeho obsah sa vždy mení. Toto rozdelenie a responzivitu stránky som zaistil pomocou frameworku Bootstrap.

Pri možnostiach jednotlivých záznamov databázy sa dajú vykonať činnosti ako upraviť, vymazať a iné. Miesto textu týchto činností sa zobrazujú ikonky, pomocou toolkitu FontAwesome[2]. Každá taká ikonka má po namierení myškou na ňu vyskakovacie okienko, ktoré vypíše o akú činnosť ide. Príklad ikonky s vyskakovacím okienkom je možné vidieť v zdrojovom kóde 5.3

```
1 <i class="fas fa-edit" data-toggle="tooltip" data-original-title="Upravit"></i>
```

Zdrojový kód 5.3: Aplikácia FontAwesome ikonky pre upravenie záznamu na HTML značku

Na zmenu štýlu stránky som použil jazyk CSS, konkrétne Less, ktorý umožňuje mať v CSS premenné. Pre všetky časti frontendu, ako farba pozadia, písma som použil premenné. Takže v prípade, že zadávateľ bude chcieť zmeniť farbu stránky stačí zmeniť farbu u približne desiatich premenných. Less sa interpretuje prehliadačom na strane klienta pomocou javascriptového súboru `less.js`, ktorý poskytuje samotná spoločnosť Less.

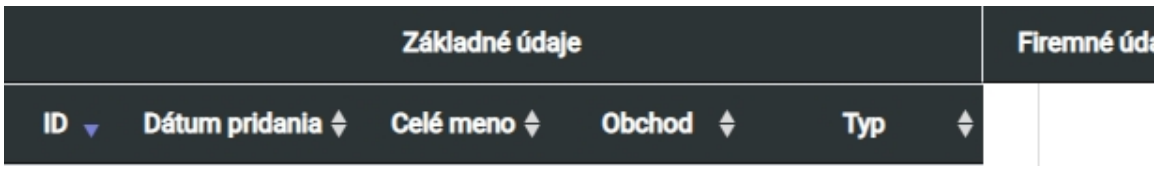
Menu zobrazené na stránke je zrealizované pomocou sidebaru. Keďže sidebar musí byť responzívny pretože môže nastať prípad kedy používateľ je na zariadení, ktorý má menšiu výšku ako je výška samotného menu. Tento prípad som vyriešil pomocou jQuery pluginu slimScroll.js[3], ktorý umožňuje listovať v menu napriek rôznej výške. Okrem tohto pluginu, sidebar ešte využíva jednu funkciu, ktorá označí kde v menu sa nachádza používateľ. Ako je možné vidieť v časti zdrojového kódu 5.4 funguje na základe URL, to znamená, že podľa URL jQuery skript priradí príslušnému HTML elementu triedu `activated`, ktorý označí štýlom to kde sa používateľ nachádza.

[2] FontAwesome - https://fontawesome.com/
[3] slimScroll - http://rocha.la/jQuery-slimScroll

```
var urlPath = window.location.pathname;
var hrefLink = urlPath.replace("/broader/","");
$("#sidebar .nav li a[href=""+hrefLink+""]").parent().addClass("activated");
```

Zdrojový kód 5.4: jQuery skript na priadenie triedy v sidebare podľa URL

Na zobrazenie údajov som použil tabuľky, vytvorené pomocou DataTables, ktoré umožňujú responzivitu, vyhľadávanie a filtrovanie. Najnovšia verzia DataTables 1.10.17 rieši všetky problémy responzivity okrem jedného. Ak je v hlavičke tabuľky `<thead>` viac ako jeden riadok `<tr>`, tak pri zmene šírky sa prvý riadok tabuľky neskryje. Problém je znázornený na obrázku 5.1.



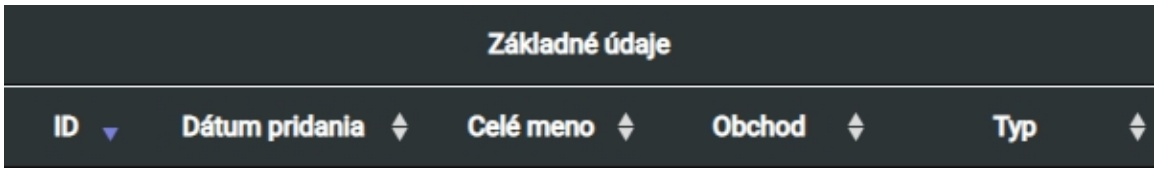
Obr. 5.1: DataTables - responzívny problém pri hlavičke tabuľky

Tento problém riešila stará verzia DataTables 1.9.4, ktorá ale nerieši priveľa chýb, ktoré naopak verzia 1.10.17 rieši. Tým pádom som nemohol použiť staršiu verziu DataTables a preto som napísal jQuery skript, ktorý tento problém rieši.

```
var theadRowCount = $("#"+tableID+" thead tr").length;
if (theadRowCount == 2) {
    var colspan = $("#"+tableID+" thead tr:nth-child(1) th:nth-child(1)").attr("colspan"
        );
    table.on("responsive-resize.dt", function(e, datatable, columns) {
        var numOfVisibleTh = $("#"+tableID+" tr.responsiveth th:visible").length;
        if (numOfVisibleTh <= colspan) {
            $("#"+tableID+" thead tr:nth-child(1) th:nth-child(2)").addClass("d-none");
        }
        else {
            $("#"+tableID+" thead tr:nth-child(1) th:nth-child(2)").removeClass("d-none")
                ;
        }
    });
}
```

Zdrojový kód 5.5: DataTables - skript na opravenie responzívneho problému

Vyššie uvedený skript pracuje s funkciou od DataTables, ktorá sa aktivuje vždy ak sa zmení šírka tabuľky a počet stĺpcov. Skript ďalej na základe viditeľných stĺpcov buď priradí danému elementu triedu `d-none`, čo spôsobí jeho skrytie alebo danú triedu odstráni.



Obr. 5.2: DataTables - tabuľka po aplikovaní jQuery skriptu

5.6 Časti systému a spoločné funkcie

V tejto sekcii je popísaná implementácia jednotlivých časti systému. Najprv sú popísané funkcie, ktoré využívajú všetky časti a potom sú popísané charakteristické funkcie pre danú časť.

5.6.1 Spoločné funkcie

Keďže každá časť systému využíva databázu, je nutné spomenúť triedu `DB`. Trieda `DB` obsahuje konštruktór ako je možné vidieť v zdrojovom kóde 5.6, ktorý sa pomocou rozhrania PDO[4] pripája k databáze s údajmi uloženými v súbore `init.php`.

```
1 private function __construct() {
2     try {
3         $this->_pdo = new PDO("mysql:host=" . Config::get("mysql/host") . ";dbname=" .
              Config::get("mysql/db"), Config::get("mysql/username"), Config::get("mysql/
              password"));
4     } catch (PDOException $e) {
5         die($e->getMessage());
6     }
7 }
```

Zdrojový kód 5.6: Konštruktor triedy `DB`

Metoda triedy `getInstance()` kontroluje či už je objekt inštanciovaný. Pokiaľ je, tak vráti samotnú inštanciu, pokiaľ nie je, tak ho inštancuje, čím vytvorí pripojenie do databázy. Okrem tejto metódy obsahuje ďalšie metódy, ktoré vykonávajú operácia nad databázou. Ide o metódy `insert()`, `update()`, `get()`, `delete()`, tieto metódy slúžia na vkladanie, aktualizovanie, získanie a vymazanie záznamov z databázy.

Vo všetkých častiach existujú tabuľky zobrazujúce údaje z databázy. Každá taká tabuľka je inicializovaná jQuery skriptom `createTable()` v hlavných súboroch systému. Skript `createTable()` nad danou tabuľkou volá konštrukčnú funkciu DataTable(), ktorá z nej urobí DataTable tabuľku. To znamená že sa do tabuľky automaticky pridá vyhľadávanie, zmena poradia a stránkovanie. Filtrovanie podľa dátumu bolo pridané pomocou jQuery pluginu - DateRangePicker[5].Keďže zadávateľ vyžadoval tlač dát z tabuliek vo formátoch .pdf, .xlsx a .csv, tak do tabuliek boli implementované DataTables pluginy, ktoré to umožňujú. Vďaka tomu, že DataTables umožňuje čítať data vo formáte JSON, ktoré môžu byť získane technológiou AJAX. Som všetky dáta získané z databázy pomocou triedy `SSP`[6] a metódy `get()` dal do jedného poľa a následne pomocou PHP funkcie `json_encode` do formátu JSON. Data vo formáte JSON si tabuľky DataTables extrahujú z príslušných súboroch v prípade potreby, napríklad pri obnove tabuľky alebo keď sú inicializované.

Data, ktoré sa pridávajú do databázy alebo upravujú, sú najprv validované pomocou triedy `Validate()`. To znamená, že všetky data, ktoré sa nachádzajú vo formulároch, konkrétne v atribúte `value` HTML značky `<input>` sa validujú. Ako je možné vidieť v zdrojovom kóde 5.7, trieda validuje, tak že dostáva dvojrozmerné pole, kde prvé pole určuje ktorý `<input>` sa má validovať. Druhé pole je v tvare `identifikátor => hodnota`, pričom identifikátor je názov pravidla a hodnota je požadovaná hodnota pravidla. Pomocou tejto triedy sa dá validovať či sú data povinné, ich minimálna a maximálna dĺžka, unikátnosť v rámci tabuľke a či obsahujú len čísla.

[4]PDO je rozhranie pre prístup k databázam
[5]DateRangePicker - http://www.daterangepicker.com/
[6]SSP - trieda od DataTables, ktorá vracia data z databázy v poli

```
$validate = new Validate();
$validation = $validate->check($_POST, array(array(
    "username" => array(
        "name" => "Prihlasovacie meno",
        "required" => true,
        "min" => 2,
        "max" => 20,
        "unique" => "user"
    ),
)
```

Zdrojový kód 5.7: Validácia prihlasovacieho mena pri vytváraní používateľa

Pridávanie a upravenie záznamov do databázy je riešené pomocou technológie AJAX a metód `create()`, `update()` v triede `DB`. Všetky formuláre pre jednotlivé kategórie sú umiestnené v priečinku `pages`. Každý taký formulár obsahuje HTML značku `<button>`, ktorý inicializuje jQuery skript `createAjaxForm()`, ktorý vykonáva asynchrónnu HTTP (Ajax) požiadavku, ako je možné vidieť v zdrojovom kóde 5.8. HTML značka `<button>` obsahuje atribút `action` špecifikujúci, ktorému súboru budú odoslané dáta z formulára na spracovanie. Takéto súbory sa nachádzajú v priečinku `form`, kde sa vykonávajú spomínané metódy.

```
function createAjaxForm(elementId) {
    var formId = $(elementId);
    formId.on("submit", function(event) {
        notification("loading");
        var req = $.ajax({
            url: formId.attr("action"),
            type: "POST",
            data: formId.serialize(),
            dataType: "json"
        });
        req.done(function(data){
            // notification and reload
        })
    });
}
```

Zdrojový kód 5.8: jQuery AJAX skript

Vymazanie záznamov z databázy je riešené podobne ako pridávanie a upravenie. Rozdiel je v inom jQuery skripte - `advancedAjax()`, ktorý vykonáva AJAX požiadavku. Tento skript je rozdielný v dvoch veciach oproti prvému skriptu - `createAjaxForm()`. Prvá je, že umožňuje odoslať parameter súboru, ktorý spracuje dáta. Druhá je, že otvorí vyskakovacie okienko, kde má používateľ možnosť potvrdiť vykonávanú činnosť alebo ju zrušiť. Pri potvrdení sa odošlú dáta súboru na ich spracovanie z formuláru spolu s parametrom, ktorý je väčšinou primárny kľúč tabuľky na špecifikovanie daného záznamu.

Vo všetkých súboroch v priečinku `form`, ktoré slúžia na pridanie, upravenie a vymazanie záznamu, sa kontroluje či naozaj ide o AJAX požiadavku. Kontrola prebieha na základe hlavičky `X-Requested-With` s hodnotou `XMLHttpRequest` ako je možné vidieť nižšie.

```
if (!$_SERVER["HTTP_X_REQUESTED_WITH"] == "XMLHttpRequest") {
    Redirect::to("index");
    exit; }
```

V prípade, že hlavička je nesprávna, nastane presmerovanie na `index`.

Pri všetkých činnostiach, ktoré sú vykonané používateľom nad databázou pomocou technológie AJAX informujú notifikáciou daného používateľa. Notifikácie sú vykonávané pomocou jQuery pluginu Lobibox, ktorý je spúšťaný jQuery skriptom - `notificaton()`. Skript dostáva dáta vo formáte JSON, kde identifikátor je buď typ správy alebo samotná správa, ktorú má vypísať. Typ správy môže byť úspech alebo nejaká chyba. Notifikačné okienko o spracovávaní požiadavky sa objaví hneď po tom čo používateľ, vykonal nejakú činnosť. Tento prípad zobrazuje zdrojový kód 5.8 na štvrtom riadku. Notifikačné okienko je otvorené dokým požiadavka nie je spracovaná. V momente, keď sa spracuje vypíše používateľovi notifikáciu o jej dokončení alebo v prípade chyby vypíše konkrétnu chybu. Okrem notifikácií sa ešte činnosti vykonané v rámci systému zaznamenávajú. Zaznamenávanie je spravené pomocou triedy `Logger()`. Zaznamenáva sa používateľ, ktorý vykonal činnosť, v aký čas a popis danej činnosti.

5.6.2 Používatelia systému

Táto časť systému bola prvá, ktorú som implementoval. Rozdelenie práv som spravil na základe use-case diagramov. To znamená, že používateľ, ktorý je neprihlásený má prístup len k stránke `login`, kde má možnosť sa prihlásiť. Údaje zadané pri prihlasovaní sa validujú, ak sú validné, používateľ je prihlásený a má prístup do systému. Pri prihlasovaní má používateľ ešte možnosť zaškrtnúť aby si zapamätal prihlásenie pri ďalšej návšteve. Túto funkčnosť som implementoval pomocou premennej `$_COOKIE[hash]`, do ktorej som uložil náhodne vygenerovaný reťazec(hash). Tento hash som ešte uložil do tabuľky `user_session` spolu s id používateľa, ktorý sa prihlasuje. Takže pri ďalšej návšteve sa porovná hash uložený v `$_COOKIE[hash]` a v tabuľke `user_session`, v prípade zhody je používateľ na základe id prihlasený.

Odhlásenie používateľa vykonáva metóda `logout()`, ktorá vymaže na strane klienta uložený cookie a na strane klienta uložený session. Pokiaľ používateľ zaškrtol aby si systém pamätal prihlásenie, vymaže sa aj záznam z tabuľky `user_session` patriaci danému používateľovi.

Používateľské práva sa kontrolujú pomocou metódy `hasPermission()` v triede `User`. Metóda `hasPermission()` pracuje s tabuľkou `groups` a s hodnotou stĺpca `ugroup` v tabuľke `user`. Metóda zistí akú má hodnotu používateľ v stĺpci `ugroup` a na základe tej hodnoty sa pozrie do príslušného záznamu tabuľky `groups`, kde zisťuje aké má používateľ práva. Vďaka tejto metóde som implementoval používateľské práva podľa use-case diagramov jednotlivým kategóriám systému.

Pri vytváraní nového používateľa sa validuje aj unikátnosť používateľského mena. To znamená, že v rámci stĺpca `username` v tabuľke `user` nesmú byť dve rovnaké používateľské mená. Po úspešnej validácii sa pomocou metódy `insert()` vložia dáta z formuláru do databázy. Všetky dáta sa vložia do databázy v normálnej forme, ako boli zadané okrem hesla, ktoré je šifrované kvôli bezpečnosti. Šifrovanie hesla som implementoval pomocou funkcie `hash()`[7], ktorú ponúka PHP od verzie 5.1.2. K heslu som pridal náhodne vygenerovaný reťazec(salt), ktorý sa zapíše za heslo a novo vzniknutý reťazec sa zašifruje algoritmom `sha256`[8]. Z tohto dôvodu je nutné ukladať do databázy aj `salt`, ktoré sa používa pre validáciu hesla pri prihlasovaní.

[7] hash - https://www.php.net/manual/en/function.hash.php
[8] sha256 - https://en.wikipedia.org/wiki/SHA-2

5.6.3 Evidencia skladu

Pri dodávateľoch a členení zásob sa využívajú metódy `insert()` `update()` `delete()`, ktoré už sú spomenuté v podkapitole 5.6.2. Čo je špecifické pre túto časť systému je sklad zásob, kde môže používateľ vytvárať objednávky. Pri vytváraní novej objednávky používateľ špecifikuje informácie o zásobe, ako dodávateľa, kategóriu, podkategóriu, odrodu, množstvo a predajnú cenu. Týchto zásob môže byť v jednej objednávke viac. To som implementoval pomocou jQuery skriptu `createAjaxGet()`, ktorý je inicializovaný po kliku na „Pridať ďalšiu položku“ pri vytváraní objednávky. Funguje na princípe metódy `get()` pomocou technológie AJAX. Dáta, ktoré predstavujú ďalšiu zásobu sú vložené na stránku za poslednú existujúcu zásobu pomocou `appendTo()`[9]. Pri každej takej zásobe existuje `<button>`, ktorý vykonáva jQuery `remove()`[10], čím vymaže element obsahujúci zásobu z dokumentu.

Informácie o jednotlivých zásobách sa ukladajú do polí, ktoré sa po pridaní objednávky postupne prechádzajú a zároveň validujú. Po úspešnej validácií sa jedna za druhou pridá do databázy pomocou metódy `create()`, ktorá predáva údaje metóde `insert()` vo forme `stĺpec:hodnota`. Pridané zásoby sa zapisujú do tabuľky `storage_supplies` spolu s id objednávky, ku ktorej patria. Okrem toho, že objednané množstvo sa zapisuje do tabuľky `storage_supplies`, zapisuje sa ešte aj do stĺpca `amount` v tabuľke `item`, kde sa uchováva celkový počet konkrétnej odrody.

V tabuľke, ktorá uchováva jednotlivé objednávky - `storage_supply_order`, existuje stĺpec - `lock_in`, ktorý definuje kedy je možné vymazať objednávku. Môže nadobúdať dve hodnoty:

- `0` – je možné vymazať objednávku.
- `1` – znamená, že objednávka je uzamknutá a nedá sa vymazať.

Po vytvorení objednávky má každá hodnotu nula, to kedy objednávka nadobudne hodnotu jedna je popísané v podkapitole 5.6.4.

Okrem stĺpca `lock_in` je nutné ešte stĺpec `approved`, ktorý definuje či je objednávka schválená alebo neschválená. Používateľ môže neschválenú objednávku schváliť. Funkciu schválenia objednávky som implementoval pomocou jQuery skriptu `advancedAjax()`, ktorý je spomenutý v podkapitole 5.6.2.

V sklade zásob je možné si pozrieť prehľad zásob, ktorý je implementovaný pomocou DataTable. Data sa importujú pomocou technológie AJAX z databázového pohľadu, ktorého tvoria tabuľky `category`, `subcategory` a `item`.

```
1 CREATE VIEW storage_supply AS
2 SELECT i. * , s.id AS subcategory_id, s.category, s.name AS subcategory_name, c.id AS
      category_id, c.name AS category_name
3 FROM subcategory s
4 JOIN item i ON i.subcategory = s.id
5 JOIN category c ON c.id = s.category
```

Zdrojový kód 5.9: Pohľad pre prehľad zásob

Zdrojový kód 5.9 zobrazuje SQL kód, ktorý som použil na vytvorenie pohľadu v rozhraní phpMyAdmin.

[9] appendTo - http://api.jquery.com/appendto/
[10] remove - https://api.jquery.com/remove/

5.6.4 Evidencia výroby

Táto časť systému je zameraná na implementáciu všetkého v rámci evidencie výroby. Podobne ako pri evidencii skladu aj táto časť využíva rovnaké metódy na pridávanie, upravovanie a vymazanie. Čo je jedinečné pre túto časť je pri vytváraní nového produktu.

Vytváranie nového produktu sa skladá z piatich častí základné informácie, náklady, použité materiály, zamestnanci pri výrobe a ostatné náklady, čo sa v ktorých nachádza je popísane v podkapitole 2.4.2. Všetky údaje sú buď text, číslo alebo dátum. V prípade dátumu používateľ vyberá dátum z kalendára, to som implementoval pomocou jQuery pluginu `dateRangePicker()`.

Keďže pri vytváraní produktu môže byť viac zamestnancov, nákladov a použitých materiálov, tak musí existovať možnosť ich pridať. Toto som implementoval podobne ako pri sklade zásob a to pomocou jQuery skriptu `advancedAjaxGet()`.

Predtým ako sa zadané údaje odošlú do databázy, prechádzajú dvojfázovou validáciou.

1. **Validácia zadaných údajov** – V prvej fáze sa postupne validujú zadané údaje pomocou triedy `Validate`, ako či je údaj povinný alebo jeho dĺžka.

2. **Validácia použitých materiálov** – Ak prvá fáza bola úspešná, tak sa prechádza na druhú fázu. V druhej fáze sa validujú použité materiály pre vytvorenie nového produktu. Úlohou tejto validácie je aby sa pri výrobe produktu nepoužilo viac materiálov ako je na sklade zásob. Túto funkčnosť som implementoval nasledovne. Najprv som skontroloval či v použitých materiálov existujú nejaké duplikácie zásob na základe ich id. Na kontrolu duplikácií som použil funkciu zobrazenú v zdrojovom kóde 5.10, ktorá vracia pole vo formáte `kľúč:hodnota`. Kľúč je číslo poradia zásoby a hodnota je id konkrétnej odrody.

```
1    array_intersect($supplyInputs, array_unique(array_diff_key($supplyInputs,
         array_unique($supplyInputs))));
```

Zdrojový kód 5.10: Funkcia na zistenie duplikácií v poli

   Ak neboli nájdené žiadne duplikácie skontroluje sa použité množstvo s množstvom na sklade zásob. V prípade ak boli nájdene duplikácie, tak nad polom, ktoré je výstupom funkcie v zdrojovom kóde 5.10 sa prevedie funkcia `transformArray()`. Hlavnou úlohou funkcie `transformArray()` je, že všetky zásoby, ktoré majú rovnakú odrodu obalí ďalším polom. Takýmto spôsobom sa uchová číslo zásoby o ktorú sa jedná, vďaka čomu zistím aké bolo použité množstvo. Použité množstvo sa spočíta a porovná s naskladneným množstvom. Pokiaľ je použité množstvo pri výrobe produktu menšie ako je na sklade zásob, validácie bude úspešná.

Po úspešnej validácií sa vykonajú následné činnosti:

1. Data, ktoré boli zadané pri výrobe sa pomocou metódy `insert()` vložia do databázy.

2. Pomocou metódy `update()` sa v databáze odráta od jednotlivých zásob v tabuľke `storage_supplies` množstvo, ktoré sa použilo pri výrobe produktu.

3. Upraví sa celkové množstvo v tabuľke `item` na základe použitého množstva pri výrobe v tabuľke.

4. Na základe použitých zásob sa zistí, ktorým objednávkam nastaviť hodnotu stĺpca `lock_in` na 1.

V zozname produktov má používateľ možnosť buď vymazať produkt alebo pozrieť si jeho detail. V detaile produktu je sú zobrazené všetky údaje o produkte, ktoré boli zadané pri jeho vytváraní. Okrem toho používateľ má možnosť si vytlačiť tieto informácie. Tlač je implementovaná pomocou jQuery pluginu - printThis[11], ktorý umožňuje tlačiť informácie len z špecifikovanej triedy v HTML značke.

Produkty, ktoré už boli použité pri predaji sa nedajú vymazať. Kontrola či už boli použité pri predaji je implementovaná, tak že sa porovná hodnota v stĺpcoch `amount_save` a `amount`, kde prvý stĺpec predstavuje vyrobené množstvo a druhý aktuálne množstvo. Ak ich počet nie je rovnaký, tak vymazať daný produkt nie je možné.

Pokiaľ používateľ vymazal nejaký produkt, tak sa nad databázou pre produkt, ktorý sa maže, vykonajú nasledovné činnosti:

1. Materiály, ktoré sa použili pre vytvorenie nového produktu sa postupne budú prechádzať a použité množstvo sa pripočíta k príslušnej zásobe do stĺpca `amount` v tabuľke `storage_supplies`. Popri tom sa ešte použité množstvo pripočíta aj k celkovému množstvu, ktoré sa zobrazuje stĺpec `amount` v tabuľke `item`.

2. Pomocou metódy `delete()` sa vymaže produkt z tabuľky `manufacture_product` a tiež sa vymažú informácie o ňom z tabuliek s prefixom `manufacture_product`.

3. Postupne pre každý materiál, ktorý sa použil pri výrobe sa na základe ich id prejde celá tabuľka `storage_supplies`, kde sa skontroluje výskyt id použité materiálu. Pokiaľ sa id v tabuľke `storage_supplies` nenachádza, tak príslušnej objednávke, ktorej patrí daná zásoba sa zmení hodnota stĺpca `lock_in` z nula na jedna.

Podobne ako pri evidencii zásob, aj tu je prehľad zobrazujúci všetky vyrobené produkty, ku ktorému má prístup každý používateľ systému. Je vytvorený z dvoch tabuliek `product_category` a `manufacture_product`.

5.6.5 Evidencia predajov

Posledná a zároveň najrozšírenejšia časť systému je práve evidencia predajov z dôvodu, že je závislá na všetkých ostatných. Pretože každý obchod obsahuje rôzne zľavy na jednotlivé kategórie produktov, tak pri vytváraní nového typu obchodu sa vykonajú dve činnosti nad databázou. Prvá je, že po vytvorení obchodu sa pomocou metódy `insert()` vloží nový obchod do tabuľky `store`. Druhá činnosť vloží do tabuľky `store_discount` všetky kategórie ku novo vytvoreného obchodu. Do tabuľky `store_discount` sa ukladajú záznamy, ktoré tvoria všetky id kategórii produktov spolu s id vytvoreného obchodu. Pre každý taký záznam sa aj ukladá percentuálna zľava v stĺpci `discount`, ktorá je po vytvorení obchodu nastavená na nula percent pre každú kategóriu produktu. Túto percentuálnu zľavu je možné meniť v detaile jednotlivých obchodov, ktorá je implementovaná metódou `update()`. Ak používateľ vymaže nejaký obchod, vymažú sa aj záznamy z tabuľky `store_discount`.

Po pridaní novej kategórie produktov sa okrem pridania danej kategórie do tabuľky `product_category` ešte pridajú záznamy do tabuľky `store_discount` pre každý existujúci obchod s prednastavenou zľavou nula percent. Po úspešnom pridaní je automaticky presmerovaný na zoznam obchodov pomocou triedy `Redirect`, ktorú som implementoval a použil pre všetky presmerovania v rámci stránky. Okrem presmerovania je vytvorená

[11] printThis - https://jasonday.github.io/printThis/

nová `$_SESSSION` premenná, ktorá inicializuje jQuery skript `notification()`. Inicializovaný skript informuje používateľa aby pre novo vytvorenú kategóriu nastavil zľavu v jednotlivých obchodoch. Po vypísaní notifikácie je `$_SESSSION` premenná vymazaná.

Jednou z najdôležitejších kategórií sú klienti. Pri pridávaní klienta používateľ vyberá typ obchodu zo `<select>` listu, do ktorého patrí a ešte vyberie typ klienta. Na základe toho aký typ klienta vyberie sa zobrazia ďalšie údaje o klientovi, ktoré musí vyplniť. Zobrazenie ostatných údajov je implementované pomocou jednoduchých jQuery funkcií `addClass()` a `removeClass()`, ktoré do HTML značky pridávajú triedu prednastavený štýl Bootstrapom `d-none`. Vloženie nového klienta do databázy je implementované ako všetky ostatné a to pomocou triedy `insert()`.

Po pridaní je používateľovi automaticky pridaný status „Aktívny“. Takéto statusy môže používateľ pridávať v ľubovoľnom počte. Pri vytváraní nového statusu okrem mena statusu, musí vybrať ešte jeho farbu. Vyberanie farby je implementované pomocou jQuery pluginu Boostrap Colorpicker[12], ktorý sa ukladá do databázy vo forme `rgb(255,255,255)`. Pridané statusy môže používateľ svojim klientom pridávať alebo upravovať v detaile klienta.

Riešenie presunu klientov z používateľa na iného používateľa v prípade ak zamestnanec už nepracuje v spoločnosti. Je dostupné v administrácií klientov, kde má prístup jedine administrátor systému. Implementoval som to, tak že pri všetkých záznamoch v databáze, ktorí patrili bývalému zamestnancovi sa v stĺpci `user_id` zmení id na iného používateľa respektíve zamestnanca.

Ústredným bodom celého systému sú predaje, ktoré sú závislé na všetkých vyššie spomenutých kategóriach. Používateľ pri zadávaní novej objednávky vyberá klienta z listu a dátum, ktorý je implementovaný pomocou jQuery pluginu - DateRangePicker. Po zadaní klienta, používateľ vyberá kategóriu produktov, ktorá inicializuje dve skripty:

1. Prvý skript je čisto implementovaný v jQuery, ktorý zaisťuje, že v liste produktov sa zobrazia len produkty patriace do vybranej kategórie.

2. Druhý skript zistí zľavu pre vybratú kategóriu pomocou technológie AJAX, ktorá odosiela id vybraného klienta a kategórie produktu súboru na zistenie zľavy.

Po vybratí produktu z listu sa tiež inicializujú dve skripty:

1. Prvý skript zobrazí ďalšie dve polia, množstvo a predajná cena.

2. Druhý skript podobne ako zisťovanie zľavy, aj tento je implementovaný pomocou technológie AJAX. Skript odosiela id vybratého produktu súboru, ktorý zistí z tabuľky `manufacture_product` aká je nastavená predajná cena.

Po zistení predajnej ceny a zľavy sa obe údaje zapíšu do `<input>`, ktoré môže používateľ meniť. Posledný údaj, ktorý používateľ je množstvo. Po zadaní množstva sa inicializuje skript, ktorý spočíta sumu a DPH.

Keďže v jednej objednávke môže byť viac predávaných produktov, tak aj tu je `<button>`, ktorý má podobnú funkcionalitu ako pri objednávke zásob spomenutej v podkapitole 5.6.3. V tomto prípade jeho funkcionalita spočíva v pridaní ďalšieho produktu na predaj. To znamená, že po kliknutí na `<button>` sa inicializuje jQuery skript `createAjaxGet()`, ktorý pomocou technológií AJAX pridá ďalší produkt. Rovnako ako pridávať, používateľ má možnosť vymazať produkty, ktoré sprostredkuváva jQuery skript - `remove()`.

Predtým ako sa zadané dáta odošlú do databázy, sa validujú dvojfázovou validáciou pomocou triedy `Validate`.

[12]Bootstrap Colorpicker - https://farbelous.io/bootstrap-colorpicker/v2/

1. **Validácia údajov** – Prvá fáza skontroluje správnosť zadaných údajov. Ak bola prvá fáza úspešná prechádza na druhú fázu.

2. **Validácia množstva** – Druhá fáza spočíva vo validácii použitého množstva. To znamená, že sa zisťuje či použitého množstva je menej alebo rovnako, ako je momentálne na sklade produktov pre daný produkt. Môžu nastať dve prípady. Obe prípady pracujú na základe výskytu duplikátov jednotlivých produktov, ktoré sa zisťujú pomocou funkcie zobrazenej v zdrojovom kóde 5.10. V prípade, že nie sú žiadne duplikáty, tak sa skontroluje produkt za produktom čí použité množstvo je menej alebo rovné tomu na sklade produktov. Ak v objednávke sú duplikáty, tak sa uložia vo forme `kľúč:hodnota`. Kľúč je číslo poradia produktu a hodnota je id predávaného produktu. Tieto duplikáty sa pomocou funkcie `transformArray()` prevedú do dvojrozmerného poľa. To znamená, že všetky produkty s rovnakým id sú obalené ďalším polom. Pre každé také vzniknuté pole sa na základe zisteného id produktu v tabuľke `manufacture_product` zistí aktuálny počet množstva na sklade, ktorý sa porovná so súčtom použitého množstva pri predaji.

Po úspešnej validácií sa zadané dáta pri vytváraní novej objednávky pomocou metódy `insert()` vložia do databázy. Samotná objednávka sa uloží do tabuľky `sale` a predané produkty, ktoré identifikuje id objednávky sa uložia do tabuľky `sale_products`. Okrem toho sa pripočíta jednotka k číslu v stĺpci `orders`, ktorá v tabuľke `client` predstavuje koľkokrát klient objednával. Podobne sa pripočíta jednotka k číslu v stĺpci `sale`, ktorý sa nachádza v tabuľke `user` a predstavuje počet objednávok, ktoré používateľ vytvoril.

Pri upravení existujúcej objednávky sa najprv množstvo jednotlivých predaných produktov, naskladní späť do tabuľky `manufacture_product` v stĺpci `amount`. Následne sa vymažú záznamy o predaných produktov z tabuľky `sale_products`. Po vymazaní produktov skript pokračuje presne ako pri pridávaní, jednotlivé údaje musia prejsť dvojfázovou validáciou a ak prejdú, tak sa vložia do databázy pomocou metódy `insert()`. V prípade ak sa pri upravovaní objednávky zmenil klient, tak klientovi, ktorý tam bol pôvodne, sa odráta jednotka od počtu objednávok v stĺpci `orders` a novému klientovi pripočíta.

Ak používateľ vymaže objednávku vykoná sa presný opak toho čo pri vytvorení. To znamená, že sa z tabuľky `sale` vymaže príslušný záznam patriaci objednávke. Na základe id objednávky sa vymažú aj jednotlivé produkty z tabuľky `sale_products`. Nakoniec sa odráta jednotka od počtu objednávok pri klientovi v tabuľke `client` a pri používateľovi v tabuľke `user`.

Z každej vytvorenej objednávky je možné vytvoriť dodací list. Tvorbu dodacieho listu som implementoval pomocou knižnice PhpSpreadsheet[13], ktorá poskytuje množstvo tried umožňujúce čítať a písať do formátov, ako .xlsx. Údaje sa vkladajú do dodacieho listu, ktorý mi poskytol zadávateľ. Pomocou knižnice PhpSpreadsheet som poskytnutý dodací list načítal a následne som ho pomocou metódy `getActiveSheet()` otvoril. Do dodacieho listu sa vkladajú všetky produkty, ktoré sa použili pri predaji získané z tabuľky `sale_products` na základe id objednávky pomocou metódy `setCellValue()`. To znamená, že sa do dodacieho listu zapíše kategória produktu, predané množstvo, predajná cena a percentuálna zľava. Okrem všetkých údajov ohľadom produktov sa ešte do dodacieho listu zapíše aktuálny dátum a číslo dodacieho listu, čo je vlastne id objednávky v tabuľke `sale`. Po zapísaní všetkých údajov sa dodací list uloží, následne sa novo vytvorený dodací list používateľovi stiahne a po úspešnom stiahnutí sa dodací list vymaže z disku aby nezaberal miesto.

[13] PhpSpreadsheet - https://github.com/PHPOffice/PhpSpreadsheet

5.7 Štatistika

Keďže štatistika vychádza z údajov zo všetkých časti systému, tak som ju implementoval ako poslednú. Štatistika sa delí na dve sekcie, hlavná štatistika a štatistika používateľa. Hlavná štatistika zobrazuje všeobecnú štatistiku v rámci celého systému. Vo všeobecnej štatistike sú zobrazene údaje, ako štatistika všetkých predajov na každý mesiac za aktuálny rok, počet predajov za posledný rok,mesiac a sedem dní, počet klientov, počet produktov na sklade a počet dodávateľov. Hlavná štatistika je implementovaná pomocou metódy `get()`, ktorá vykonáva `SELECT` nad databázovým pohľadom zobrazeným v zdrojovom kóde 5.11.

```
1  CREATE VIEW informations_view AS
2  SELECT
3      ( SELECT COUNT(*) FROM supplier) AS suppliers,
4      ( SELECT COUNT(*) FROM client) AS clients,
5      ( SELECT COUNT(*) FROM manufacture_product WHERE amount > 0) AS products,
6      ( SELECT COUNT(*) FROM sale WHERE sale_date >= DATE_SUB(NOW(),INTERVAL 1YEAR)) AS
           yearSales
7      /* rest of SELECTs for monthSales and weekSales */
```

Zdrojový kód 5.11: Časť kódu databázového pohľadu - `informations_view`

Štatistika jednotlivých používateľov zobrazuje údaje o klientoch a predajoch. Jak pri klientoch, tak pri predajoch sa zobrazujú údaje pre každý mesiac v aktuálnom roku, celkový počet, za posledných sedem dní a za posledných tridsať. Popri štatistike predajoch sa ešte zobrazuje predané množstvo a tržba. Tieto data pre jednotlivé obdobia sa získavajú pomocou funkcie `getStats()`, ktorá vykonáva `SELECT` nad tabuľkami v databáze.

V oboch sekciách štatistiky sa údaje pre každý mesiac v aktuálnom roku zobrazujú graficky pomocou grafu, ktorý je implementovaný JavaScript pluginom Chart.js[14]. Chart.js umožňuje vytvárať responzívny graf v rôznych tvaroch na základe vstupných dát. Data predávané pluginu na vytvorenie grafu sa získavajú funkciou `getMonthlyStats()`, ktorá pre jednotlivé mesiace vykonáva `SELECT` nad tabuľkou v databáze.

Okrem týchto dvoch sekciách, každý používateľ informačného systému môže pri tabuľkách, ktoré sú vytvorené pomocou DataTables vidieť v nejakej forme štatistiky, keďže umožňujú filtrovanie, vyhľadávanie a zoradenie zobrazených výsledkov. Čím vyššie sú práva používateľa, tým viac štatistík vidí. Napríklad administrátor má práva vidieť v administrácií klientov, kto akého klienta spravuje. Taktiež má práva k záznamom, kde vidí všetky činnosti vykonané používateľmi s dátumom a časom v rámci systému.

[14]Chart.js - https://www.chartjs.org/

Kapitola 6

Testovanie

Cieľom testovania je znížiť riziko výskytu problémov, ktoré môžu nastať počas prevádzky. Ďalším cieľom testovania je aj zaistenie kvality systému. Existuje veľa spôsobov ako testovať informačný systém. V mojom prípade testovanie prebiehalo mnou štyrmi a zamestnancami spoločnosti. Jedným z testerov bol manažér, ktorý mal práva administrátora, ďalšími testermi boli dvaja predajcovia a jeden správca skladu. Tým, že to testovali zamestnanci sa overilo, či systém spĺňa zadané požiadavky a taktiež ich prácu so systémom.

Počas implementácii som zároveň testoval samotný systém pomocou chrome devtools[1] a firefox developer tools[2]. V oboch nástrojoch som využíval konzolu, v ktorej som kontroloval správnu funkčnosť javascriptových a jQuery skriptov, keďže v prípade chyby sa konkrétna chyba objaví v konzole. Okrem toho konzola vypisuje aj chybu v prípade chýbajúceho súboru alebo obrázka. Správnosť zapísaných dát v databáze vykonané pridaním alebo úpravou som kontroval pomocou phpMyAdmina.

Testovanie zamestnancami prebiehalo vždy po implementácií jednej časti systému. Zamestnanci testovali podľa požiadavok, ktoré zadal zadávateľ a ešte to čo som im zadal aby skúsili. To znamená, že sa testovali funkčnosti, ako pridanie, upravenie a vymazanie pre jednotlivé kategórie v rámci implementovanej časti. Vždy po dokončení testovania jednej časti mi poslali spätnú väzbu, čo upraviť. Podobne to prebiehalo aj pri testovaní front-endu. Aké chyby sa našli, na akých zariadeniach a či boli opravené je podrobnejšie spísané v podkapitolách 6.1 a 6.2.

6.1 Testovanie front-endu

V prvom rade bol front-end testovaný mnou počas implementácií a následne zamestnancami. Testovalo sa všetko týkajúce sa front-endu. Front-end bol testovaný na troch zariadeniach, na počítači, na mobile a na tablete. Okrem rôznych zariadení sa testoval front-end aj na iných prehliadačoch ako:

- Google Chrome 73.0
- Mozilla firefox 66.0
- Opera 58.0
- Microsoft Edge 17.0

[1] Chrome DevTools - https://developers.google.com/web/tools/chrome-devtools/
[2] Firefox developer tools - https://developer.mozilla.org/son/docs/Tools

Hlavnou časťou testovania front-endu bola responzivita, ktorá sa testovala na základe zobrazených údajov či už v tabuľkách alebo všeobecne na stránke. Zamestnancom som povedal aby prešli všetky kategórie na webe a v spätnej väzbe nech mi pošlú spolu s chybami, prehliadač s jej verziou a rozlíšenie, na ktorom testovali. Jednou z chýb bolo veľmi pomalé rolovanie v sidebare na mobilných zariadeniach. Táto chyba bola opravená, keďže plugin, ktorý poskytuje rolovanie umožňuje meniť tu rýchlosť. Ďalšou chybou bola pri pridávaní, pôvodne to bolo implementované, že polia, kde sa zadávajú údaje mali len ikonka a `placeholder`, ktorý informoval o aké pole sa jedná, čo bolo pre zamestnancov nepriehľadné. Z tohto dôsledku bolo ku každému takému poľu implementovaný na konci text, ktorý popisoval o čo sa jedná ako to už bolo spravené pri upravovaní. Najväčšia chyba na front-ende bola pri tabuľkách `DataTable` zobrazujúcich klientov. Popis tejto chyby a jej riešenie je v podkapitole 5.5. Súčasťou týchto chýb boli aj textové chyby vo všetkých častiach systému, ktoré boli hneď opravené.

6.2 Testovanie jednotlivých častí systému

Ako už bolo spomenuté, vždy po implementácií jednej časti systému sa naimplementovaná časť testovala zamestnancami. Najprv sa testovala časť používatelia systému a všetko s ňou spojené. Keďže odkazy pre ostatné časti systému už boli spravené bez funkčnosti, tak sa testovali práva jednotlivých rolí. Taktiež sa testovalo zamestnancami aj pridávanie, upravovanie a vymazanie v kategóriach patriace tejto časti systému, kde bola nájdená chyba. V prehliadači firefox nefungovali `onClick` eventy, ktoré vykonávali vymazanie záznamu z databázy. Túto chybu som opravil, odstránením nadbytočného kódu, ktorý to spôsoboval.

V druhej časti systému sa testovala najmä práca so skladom, ako vytváranie objednávok a ich vymazanie, kde sa našla aj chyba. Pri pridávaní položiek a ich vymazaní sa dookola inicializoval notifikačný skript, ktorý vypisoval notifikáciu o vykonaní danej činnosti. Po zistení tejto chyby zamestnancami, som ju následne opravil, tak že notifikačný skript mohol byť inicializovaný maximálne jedenkrát. Okrem testovania práce so skladom sa ešte testovali dodávatelia, kde boli problémy len s textom, ktoré som aj následne opravil.

Pri testovaní evidencii výroby som zamestnancom zadal aby vytvorili aspoň päť produktov a následne ich vymazali. Počas vytvárania nového produktu sa kategória produktu zle zapisovala do databázy. Okrem tejto chyby, ktorá bola následne opravená, sa žiadna ďalšia chyba pri vytváraní a vymazaní produktu nenašla. Podobným štýlom sa testovali kategórie produktov. Pri vymazaní kategórie produktu zamestnancom vypísalo chybu pri tabuľke produktov, pretože sa odkazovalo na vymazanú kategóriu produktu. Túto chybu som ošetril, že v takom prípade vypíše príslušný text. Toto ošetrenie som aj následne implementoval vo všetkých DataTable tabuľkách.

Posledná implementovaná časť je evidencia predajov, ktorá sa aj ako posledná testovala. V tejto časti zamestnanci testovali tri kategórie: typy obchodov, klientov a predaje. Vo všetkých kategóriach sa testovalo pridávanie, upravenie a vymazanie existujúcich záznamov. Pri obchodoch sa ešte testovalo nastavenie zliav pre jednotlivé kategórie produktov. Pri klientoch sa testovalo pridanie oboch typov klienta, kde som na základe spätnej väzby musel zmeniť povinné údaje, ktoré sa musia vyplniť pri pridaní klienta. Najviac testovaná časť v rámci celého systému bol predaj produktov, v ktorej zamestnanci dostali úlohu vytvoriť niekoľko predajov rôznym klientom s rôznym počtom produktov a následne vytvoriť dodací list pre danú objednávku. Keďže táto časť využíva veľa funkcií z minulých častí, ktoré už boli opravené nenašli sa žiadne chyby okrem zlého zápisu údajov do buniek v dodacom liste. Po zistení tejto chyby, bola následne aj hneď opravená.

Kapitola 7

Záver

Cieľom tejto bakalárskej práce bolo navrhnúť a implementovať informačný systém pre evidenciu výroby, skladu a predaja. Takýto systém je určený najmä pre spoločnosti, ktoré si vyrábajú vlastné produkty a následne ich predávajú svojím klientom z vlastného skladu. Hlavným cieľom takéhoto systému je nielen evidovať všetky udalosti či už pri výrobe, sklade alebo predaja, ale aj export štatistík z evidovaných údajov.

Tieto ciele boli dosiahnuté v niekoľkých krokoch. V prvom rade sa analyzovali požiadavky zadávateľa, podľa ktorých sa vyhľadali existujúce riešenia s podobným zameraním. Na základe analýzy požiadavkov a existujúcich riešení bol informačný systém rozdelený na štyri časti, pre ktoré bolo nutné preštudovať a vybrať vhodné technológie. Po vybratí vhodných technológií bol spravený návrh informačného systému, podľa ktorého sa následne aj implementovalo. Návrh systému predstavovali hlavne use-case diagramy a schéma relačnej databázy, ktorá poslúžila pri implementácií databázy. Implementácia vždy prebiehala po častiach, ktoré boli vždy jednotlivo testované zamestnancami spoločnosti. Vo výsledku by sa dalo povedať, že implementovaný informačný systém je vlastne kombináciou niekoľkých existujúcich systémov s funkcionalitou navyše.

7.1 Budúci vývoj

V budúcnosti by sa dal informačný systém rozšíriť o ďalšiu časť, ktorá by predstavovala správu skladov. Momentálne je informačný systém postavený na tom, že existujú iba dve univerzálne sklady jeden pre produkty a druhy pre zásoby. Nová časť by umožňovala správcom skladu vytvárať sklady či už pre zásoby alebo produkty, presúvať produkty/zásoby medzi skladmi, pri vytváraní nového produktu definovať, do ktorého skladu sa naskladní a nakoniec pri predaji definovať z ktorého skladu sa predajú produkty. Vďaka tejto časti by vznikla aj nová kategória štatistiky, kde by sa dalo lepšie pozorovať čo sa kde naskladňuje poprípade objednáva a z ktorého skladu sa koľko a čo predáva klientom.

Ďalším vylepšením by bola úprava a rozšírenie databázy, keďže nie je dokonalý. Pomohlo by k tomu vytvorenie databázového pohľadu a vytvorenie samostatných tabuliek pre produkty a zásoby, ktoré by uchovávali iba aktuálne množstvo.

Ďalej mi napadlo rozšíriť už existujúcu časť, a to používateľov systému. Rozšírenie by spočívalo v tom aby si používatelia mohli definovať vlastné role, ktorým by mohli následne pričleniť práva. Momentálne naimplementovaná databáza už s tým počíta, ale nie je to naimplementované. V rámci tejto časti by sa dala rozšíriť aj kategória úlohy. Rozšírenie by

spočívalo aby si používatelia mohli sami vytvárať a potvrdzovať úlohy. Takéto úlohy by ale museli nejako o ich existencii informovať administrátora.

Posledné rozšírenie, ktoré mi napadlo popri implementácií je vytváranie upozornení pre klientov alebo pre jednotlivé produkty a zásoby v sklade. To znamená, že administrátor by bol schopný vybrať produkt, kedy sa pošle upozornenie, napríklad ak bude počet menší než nejaké číslo a na aký email s akým popisom sa pošle upozornenie. Podobne by to fungovalo aj pri klientoch, kde by predajca mohol vytvoriť upozornenie, ktoré by ho upozornilo napríklad v prípade pravidelnej objednávky alebo neaktivity.

Literatúra

[1] Arlow, J.; Neustadt, I.: *UML 2 a unifikovaný proces vývoje aplikací*. Computer Press, 2007, ISBN 978-80-251-1503-9.

[2] Chaffer, J.; Swedberg, K.: *Learning jQuery : Better Interaction Design and Web Developmennt with Simple JavaScript Techniques*. Packt Publishing Ltd., 2007, ISBN 978-1-847192-50-9.

[3] Gutmans, A.; Bakken, S. S.; Rethans, D.: *Mistrovství v PHP 5*. Computer Press, 2012, ISBN 978-80-251-1519-0.

[4] Hauser, M.; Hauser, T.; Wenz, C.: *HTML a CSS Velká kniha řešení*. Computer Press, a.s., 2006, ISBN 80-251-1117-2.

[5] Ľubovslav Lacko: *1001 tipů a triků pro SQL*. Computer Press, 2011, ISBN 978-80-251-3010-0.

[6] Leiss, O.; Schmidt, J.: *PHP v praxi : pro začátečníky a mírně pokročilé*. Grada Publishing, a.s., 2008, ISBN 978-80-247-3060-8.

[7] Lindley, C.: *Front-End Developer Handbook 2017*. Január 2014, [Online; 24.02.2019]. URL https://ioeduc.webitcloud.net/drive/Book/front-end-handbook-2017.pdf

[8] MySQL: *MySQL 5.0 Reference Manual*. November 2016, [Online; 25.02.2019]. URL https://downloads.mysql.com/docs/refman-5.0-en.a4.pdf

[9] Naramore, E.; Gerner, J.; Scouarnec, Y. L.; aj.: *Vytváříme webové aplikace v PHP5, MySQL a Apache*. Computer Press, 2005, ISBN 80-251-1073-7.

[10] Pecinovský, R.: *Návrhové vzory*. Computer Press, 2013, ISBN 978-80-251-1582-4.

[11] Písek, S.: *JavaScript : efektní nástroj oživení WWW stránek*. Grada Publishing, spol. s.r.o., 2001, ISBN 80-247-0014-X.

[12] Schneider, R. D.: *MySQL: oficiální průvodce tvorbou, správou a laděním databází*. Grada Publishing, 2006, ISBN 80-247-1516-3.

[13] Sellier, A.: *LESS—The Dynamic Stylesheet Language*. Január 2017, [Online; 23.02.2019]. URL http://lesscss.org/

[14] Sommerville, I.: *Softwarové Inženýrství*. Computer Press, 2013, ISBN 978-80-251-3826-7.

[15] Suehring, S.; Converse, T.; Park, J.: *PHP 6 and MySQL 6 Bible*. Wiley Publishing, Inc., 2009, ISBN 978-0-470-38450-3.

[16] TutorialsPoint: *Bootstrap tutorial*. Január 2014, [Online; 23.02.2019].
URL http://wiki.lib.sun.ac.za/images/0/07/Bootstrap-tutorial.pdf

[17] TutorialsPoint: *CSS Cascading style sheets*. Január 2017, [Online; 22.02.2019].
URL https://www.tutorialspoint.com/css/css_tutorial.pdf

[18] TutorialsPoint: *Less*. Január 2017, [Online; 23.02.2019].
URL https://www.tutorialspoint.com/less/less_tutorial.pdf

[19] Vrána, J.: *1001 tipů a triků pro PHP*. Computer Press, 2012, ISBN 978-80-251-2940-1.

[20] Wodehouse, C.: *A Beginner's Guide to Back-End Development*. Jún 2017, [Online; 25.02.2019].
URL https://www.upwork.com/hiring/development/a-beginners-guide-to-back-end-development/

[21] Zelinka, T.; Svítek, M.: *Telekomunikační řešení pro informační systémy síťových odvětví*. Grada Publishing, 2009, ISBN 978-80-247-3232-9.

[22] Žára, O.: *JavaScript : Programátorské techniky a webové technologie*. Computer Press, a.s., 2015, ISBN 978-80-251-4573-9.

Príloha A

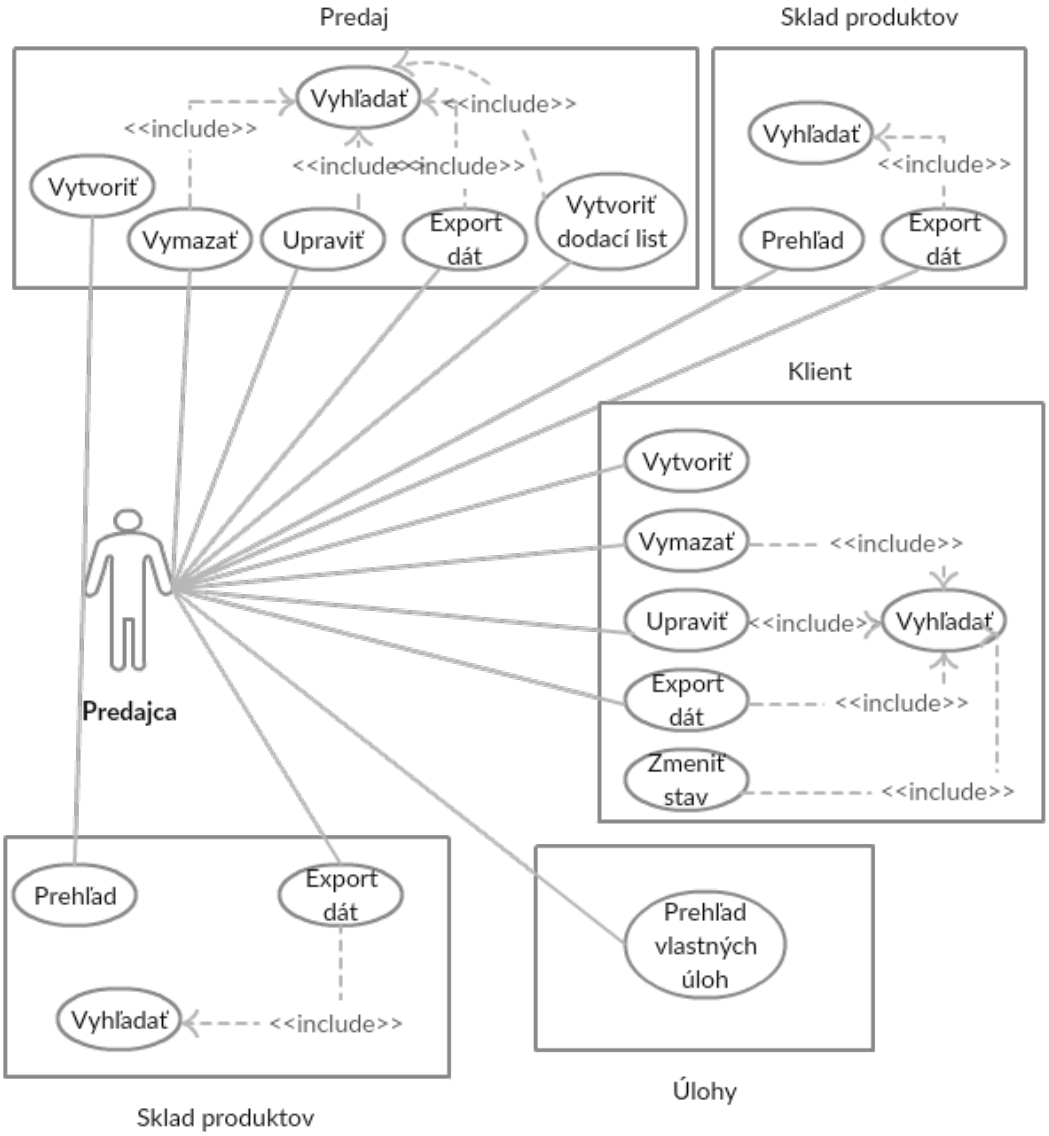
Obsah priloženého pamäťového média

Obsahom priloženého pamäťového média je:

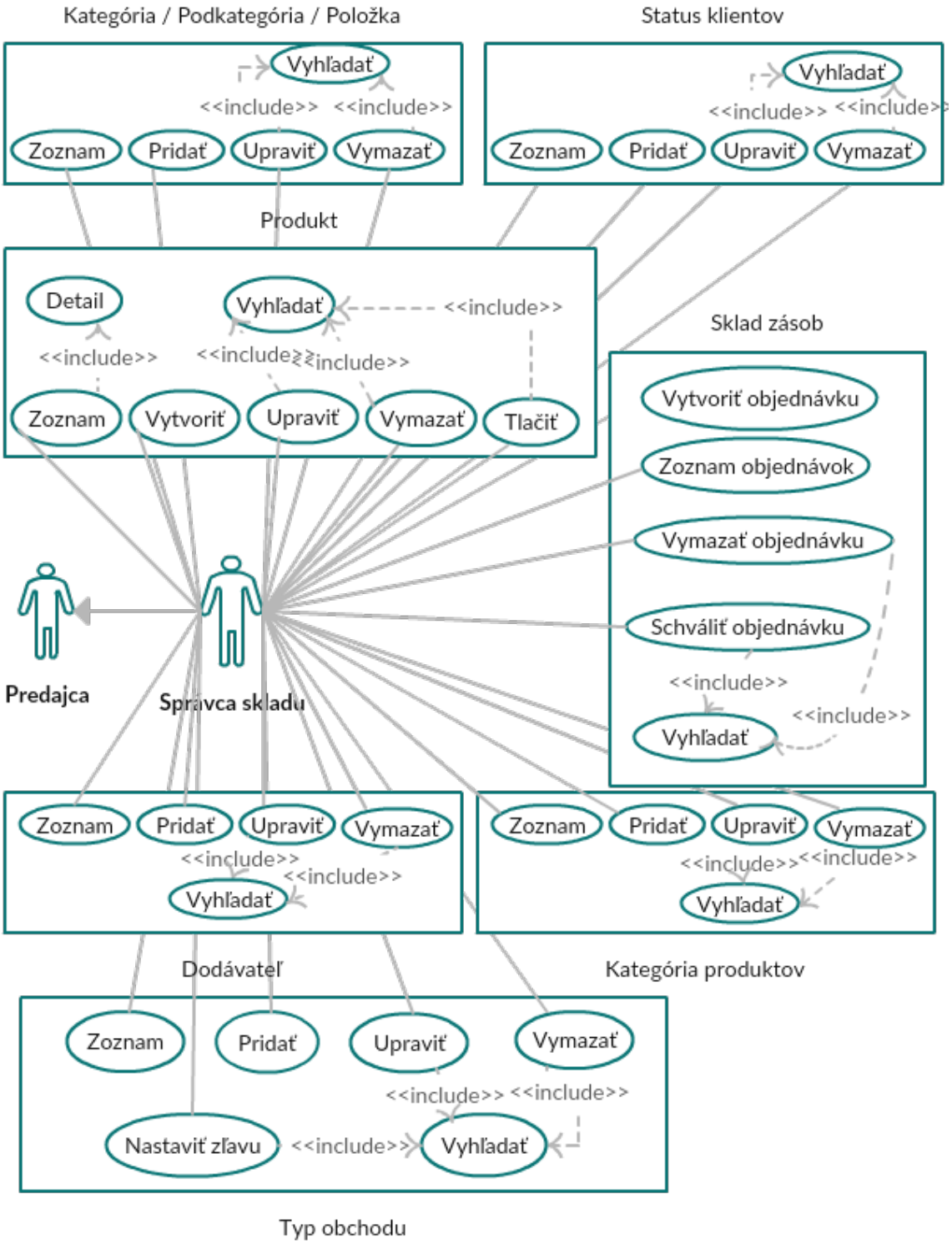
- text bakalárskej práce,
- zdrojové súbory šablony LaTeXpre vysádzanie textu bakalárskej práce,
- zdrojové súbory informačného systému,
- manuál pre zavedenie informačného systému.

Príloha B

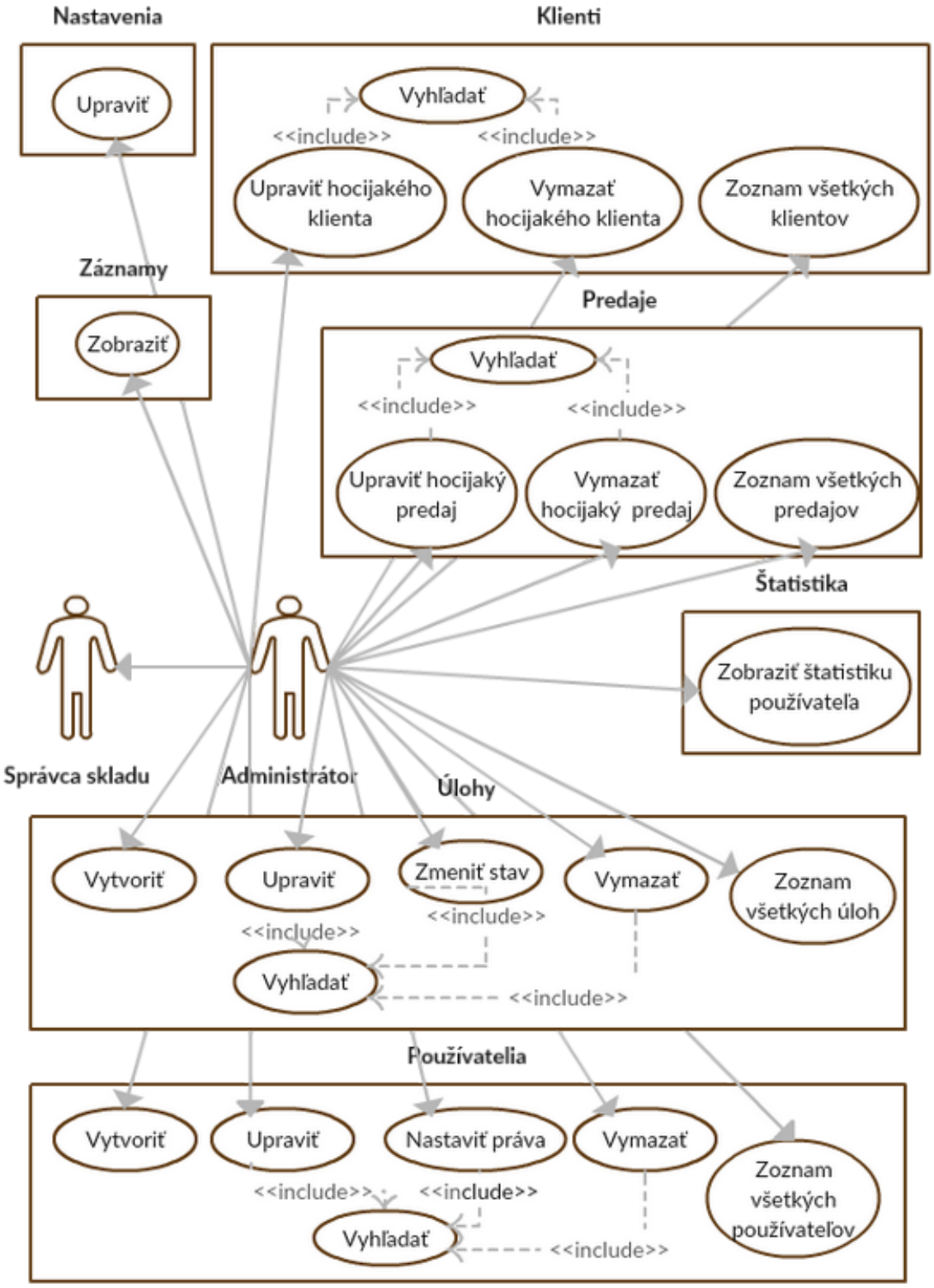
Diagramy a schéma databázy



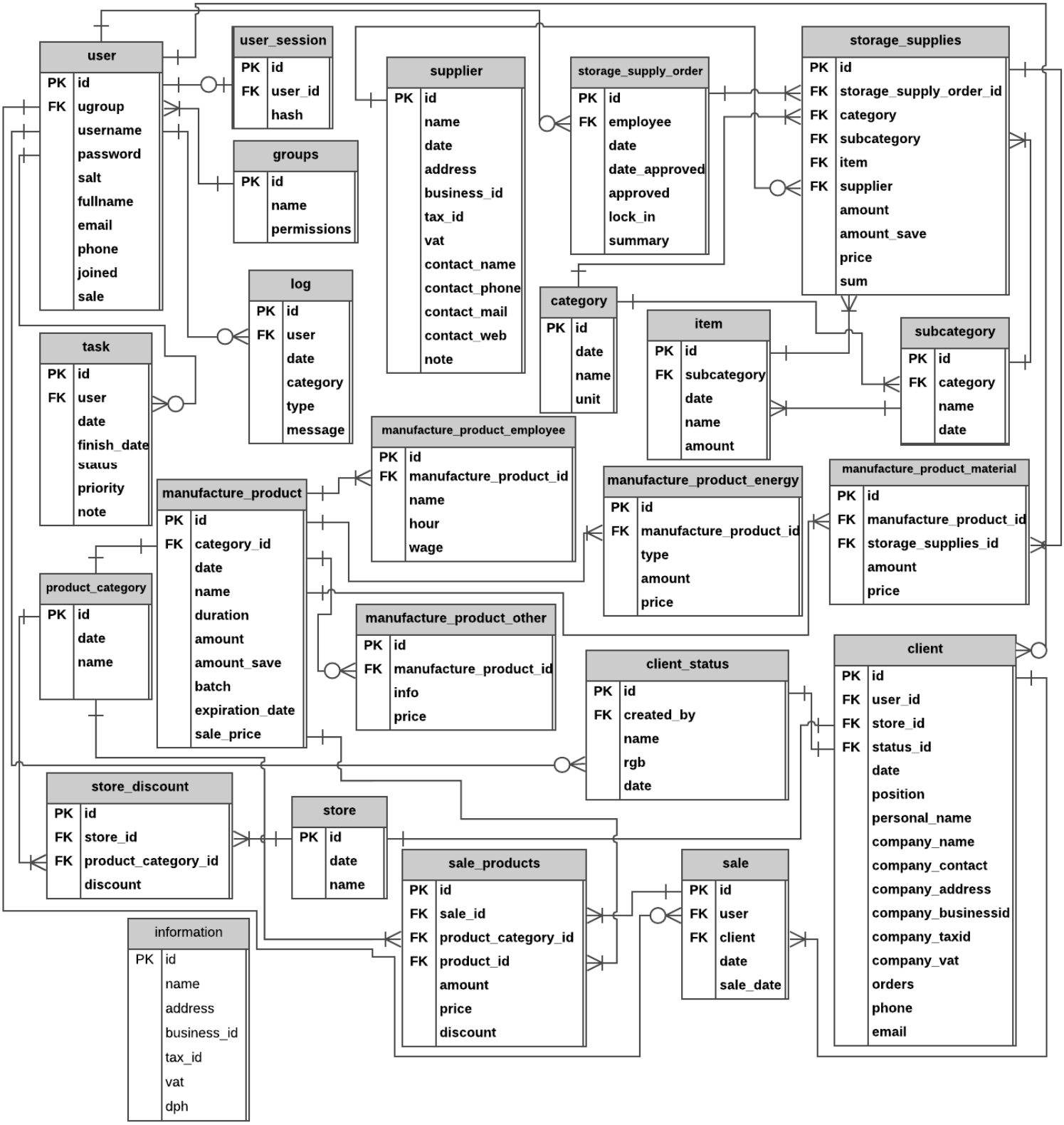
Obr. B.1: Use-case diagram aktéra - predajca



Obr. B.2: Use-case diagram aktéra - správca skladu



Obr. B.3: Use-case diagram aktéra - administrátor



Obr. B.4: Schéma relačnej databázy