VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO ŘÍZENÍ VÝROBY

INFORMATION SYSTEM FOR PRODUCTION CONTROL MANAGEMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **RICHARD SCHLÉGER**
AUTHOR

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

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Schléger Richard**
Program: Informační technologie
Název: **Informační systém pro řízení výroby**
Information System for Production Control Management
Kategorie: Informační systémy

Zadání:

1. Prostudujte současné technologie pro tvorbu informačních systémů s webovým rozhraním.
2. Seznamte se se způsobem řízení výroby a skladových zásob ve vybraných firmách a s existující softwarovou podporou.
3. Analyzujte požadavky a po dohodě s vedoucím navrhněte architekturu systému pro řízení výroby ve vybrané firmě.
4. Implementujte navržený systém pomocí vhodných technologií. Implementujte možnost exportu objednávek a statistik ve formátu XLSX pro další zpracování.
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 https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Burget Radek, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 14. května 2020
Datum schválení: 22. října 2019

Abstrakt

Cieľom tejto bakalárskej práce je návrh a následná implemetácia informačného systému pre riadenie výroby. Ide o informačný systém na webovom rozhraní, ktorý je postavený na databázovom serveri MySQL a implementovaný v jazyku Java s využitým Spring Frameworku. Vzhľad systému a jeho prezentácia používateľovi je implementovaná pomocou HTML stránok formátovaných jazykom CSS.

Abstract

The purpose of this bachelor thesis is design and following implementation of information system for production control management. It is information system on web interface, based on MySQL database server and implemented in Java using Spring Framework. System appearance and its presentation to users is implemented using HTML pages formated by CSS.

Kľúčové slová

Informačný systém, webová aplikácia, riadenie výroby, Java, Spring, MySQL, HTML, CSS

Keywords

Information system, web application, production management, Java, Spring, MySQL, HTML, CSS

Citácia

SCHLÉGER, Richard. *Informační systém pro řízení výroby.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

Informační systém pro řízení výroby

Prehlásenie

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

.....
Richard Schléger
21. mája 2020

Poďakovanie

Rád by som poďakoval vedúcemu mojej práce, pánovi Ing. Radkovi Burgetovi Ph.D. za pomoc pri riešení a za vedenie mojej bakalárskej práce. Ďalej by som rád poďakoval Ing. Miroslavovi Schlégerovi za poskytnuté informácie a odbornú pomoc pri rozbore výrobných postupov.

Obsah

Kapitola 1

Úvod

Automatizácia a informatizácia procesov sa v posledných rokoch začala rozširovať vo všetkých odvetviach, nielen vo svete informačných technológií. Nie je teda prekvapením, že v dnešnej dobe takmer každý podnik, či už malý alebo veľká výrobná prevádzka využíva aspoň v malej miere výhody tejto informatizácie. V mnohých prípadoch sa jedná práve o informačný systém, ktorý šetrí firme prostriedky nielen finančné, ale hlavne časové.

V súčasnosti sú najviac rozšírené informačné systémy na webovom rozhraní. Je to úplne pochopiteľné, pretože obsahujú veľké množstvo výhod, ako napríklad žiadna potreba inštalácie na nové zariadenia, prístup nie je obmedzený na lokalitu ani na využívaný operačný systém či zariadenie a veľa daľších. Preto aj informačný systém, ktorým sa zaoberá táto bakalárska práca je postavený na webovom rozhraní.

Táto bakalárska práca sa dá rozdeliť na dve hlavné časti - teoretickú a praktickú. Teoretická časť sa zaoberá najskôr analýzou výroby a jej riadenia popísanej v kapitole 2.1 ako aj aktuálnej softvérovej podpory 2.2 a následne analýzou techológií pre tvorbu informačných systémov, ktorá je obsahom kapitoly 4. Praktická časť sa skladá z návrhu samotného systému v kapitole 3, jeho následnej implementácie opísanej v kapitole 5 a posledná kapitola 6 sa zaoberá testovaním a jeho výsledkami.

Kapitola 2

Problematika riadenia výroby

Skúmanie problematiky riadenia výroby, analýza procesu výroby a taktiež analýza aktuálnej softvérovej podpory prebiehala vo firme Tesla Liptovský Hrádok a.s.

2.1 Analýza procesu výroby

Každá výrobná firma potrebuje pre svoj chod vhodný systém, pomocou ktorého riadi a kontroluje svoju produkciu. Celá výroba sa odvíja od požiadaviek zákazníka a jeho špecifikácií.

Reálny proces výroby začína prijatím zákazky zo strany objednávateľa. Objednávka musí obsahovať všetky informácie potrebné k realizácií výroby a požadovaný termín splnenia. Dôležitou otázkou po prijatí objednávky je, či požadovaný produkt bol už v minulosti vyrábaný, a teda je k nemu dostupná všetka potrebná materiálová a výkresová dokumentácia, alebo sa jedná o nový výrobok, pre ktorý je potrebné celú túto dokumentáciu vypracovať. Procesu vypracovania dokumentácie a výrobných postupov sa budeme venovať neskôr v tejto kapitole. Po preverení alebo prípadnom vytvorení výrobnej dokumentácie je potrebné preveriť, či máme dostatočné materiálové krytie na realizáciu kompletnej objednávky. Ďalej je nutné zvážiť výrobné kapacity na požadovaný termín dokončenia zákazky. Až po úspešnom overení týchto faktorov je možné zákazníkovi potvrdiť finálny termín dodania. Pri plánovaní finalizácie výroby musíme brať do úvahy aj čas potrebný na transport k zákazníkovi, čo v prípade zahraničných zákazníkov môže byť aj niekoľko dní.

Ako sme už spomenuli, v prípade nového výrobku je potrebné vytvoriť novú výrobnú dokumentáciu a samotný postup výroby. Toto má na starosti technické oddelenie prípravy výroby, ktoré musí presne definovať všetky parametre potrebné k výrobe, to znamená výrobné výkresy, schémy a podobne. Taktiež musí definovať presné a jednoznačné pracovné postupy a použité stroje a nástroje. Pracovný postup pozostáva z jednotlivých operácií potrebných k výrobe finálneho výrobku. Tieto operácie majú presne dané poradie činnosti a taktiež zoznam materiálu nutného k vykonaniu danej operácie, tzv. konštrukčný kusovník. Každá materiálová položka použitá vo firme musí byť identifikovaná unikátnym identifikátorom. V prípade, keď je k realizácií nového výrobku potrebný materiál, ktorý sa vo firme nikdy doteraz nepoužíval, je potrebné tento materiál do systému vložiť spolu s prideleným identifikátorom. Je dôležité, aby sa v systéme nevyskytovali duplicitné údaje pre rovnaký artikel, teda jeden materiál nemôže byť vedený pod viacerými identifikátormi. Ak napríklad rovnaký materiál je vedený pod dvoma identifikátormi X a Y, môže systém chybne vyhodnotiť nedostatok materiálu pod kódom X, pričom rovnakého materiálu s identifikátorom Y je na sklade dostatok. Toto môže mať za následok nezadanie zákazky do výroby z dôvodu

nedostatku materiálu, pričom materiálu je na sklade dostatok. Sled týchto operácií tvorí tzv. pracovný postup, ktorý musí prechádzať spolu s výrobkom celým výrobným procesom. To znamená, že ak výrobok prechádza viacerými pracoviskami, musí byť jednoznačne jasné, ktoré operácie už boli ukončené a v akom objeme. Tento postup by spravidla mal končiť operáciou balenia. Ďalšou dôležitou úlohou technického oddelenia je optimalizovať výrobné dávky tak, aby sa zefektívnilo využitie strojov a minimalizovali prestoje, spôsobené nastavovaním stroja pre daný výrobok. Preto je efektívne zlučovať rovnaké operácie do väčšieho pracovného cyklu, pretože pri tvorbe ceny sa berie do úvahy aj táto veľkosť výrobnej dávky.

Aby systém riadenia výroby fungoval správne, je nevyhnutné dodržiavať disciplínu pri odhlasovaní jednotlivých operácií. Len takto docielime aktuálnosť a pravdivosť údajov, ako sú skladové zásoby, rozpracovanosť výroby a podobne. Pri neodhlasovaní reálne ukončených operácií výroby systém nevie správne vyhodnotiť skladové zásoby. Môže nastať situácia, že do dvoch rozdielnych zákaziek vstupuje rovnaký materiál. Ak reálne dokončené operácie nebudú v systéme odhlásené a tým nebude systémovo odpísaný spotrebovaný materiál, môže systém ukazovať na sklade dostatok materiálu na realizáciu druhej zákazky, pričom tento materiál už bol spotrebovaný na prvú zákazku. Kvôli tomu je potrebné odhlasovať aj čiastkovú výrobu, napríklad pri zákazke na 1000 kusov a dennej realizácií 200 kusov je potrebné túto výrobu priebežne ohlasovať, nie až po ukončení celej zákazky. Pri odhlásení poslednej operácie je výrobok automaticky pripísaný na sklad hotových výrobkov.

V reálnej firme je potrebných viacero skladov, z ktorých každý má inú funkciu. Vstupný sklad slúži ako hlavný sklad, kde je uložený všetok materiál, ktorý sa práve nepoužíva vo výrobe. Práve na tento sklad sa pozerá pri zadávaní zákazky do výroby. Systémovo môže sklady obsluhovať jedine skladník, ktorý môže prijímať materiál od dodávateľov na vstupný sklad, vydávať a presúvať materiál na výrobné sklady a takisto expedovať produkty zo skladu hotových výrobkov. Expedícia prebieha na základe aktuálnych objednávok. Skladník vyberie objednávku, skontroluje množstvo hotových výrobkov a vygeneruje sprievodný doklad ku expedovaným produktom. Pri tomto generovaní sa automaticky zníži aktuálne množstvo hotových výrobkov na sklade o expedované množstvo. Pokiaľ bola vyexpedovaná celá objednávka, ekonóm v zozname objednávok vidí, že objednávka je kompletne vyrobená a pripravená na expedíciu a môže ju uzavrieť a vystaviť faktúru zákazníkovi.

2.2 Aktuálna softvérová podpora

Aktuálna softvérovú podporu zabezpečuje firma SOFTIP so sídlom v Banskej Bystrici s ich pruduktom PROFIT. Jedná sa o informačný systém s potrebou inštalácie na každé zariadenie, kde ho chceme využívať. Primárne zameranie tohto informačného systému je ekonomická stránka riadenia, v ktorej plne pokrýva požiadavky. Zaostáva však v riadení výrobných procesov, skladov a plánovaní výroby. Za účelom riadenia výroby bol tento systém použitý prvýkrat práve vo firme Tesla Liptovský Hrádok a.s., kde táto analýza prebiehala.

Ekonomická časť systému SOFTIP PROFIT funguje na požadovanej vysokej úrovni. Správne prepočítava ceny výrobkov s ohľadom na výrobné a materiálové náklady, v poriadku generuje faktúry a aj ich archivuje, taktiež generuje potreby nákupu materiálu na základe jeho spotreby a skladových zásob. Cieľom tejto práce je však práve časť systému zodpovedná za riadenie výroby a skladov, v ktorej má aktuálny systém veľké medzery.

Systém v aktuálnej verzií nedokáže generovať plánovanie do reálnych kapacít výroby. To znamená, že neberie do úvahy strojové a ľudské kapacity podniku. Rovnako vopred neoveruje skladové zásoby, to znamená, že do výroby zaradí aj produkt, ktorý nie je krytý materiálom. Z toho môžu vo výrobe vznikať problémy, že finálny produkt nie je možné

dokončiť z dôvodu nedostatku materiálu. Z hľadiska užívateľského prístupu je tento systém pre bežného užívateľa veľmi zložitý, keď berieme do úvahy, že jednotlivé operácie výroby má každý robotník po dokončení v systéme odhlásiť kvôli aktuálnosti údajov (teda stav skladov a rozpracovanosť výroby). Po zavedení tohto informačného systému do ostrej prevádzky nastal výrazný pokles objemu výroby práve z dôvodu, že pracovníci veľkú časť pracovnej doby strávili odhlasovaním výroby. Na ohlásenie jedného výrobku v tomto systéme je potrebné vykonať veľa čiastkových odhlásení. Situácia sa postupne vylepšila komplexným zaškolením operátorov výroby a zavedením možnosti hromadného odhlasovania. Práve v tomto sa snaží byť mnou navrhovaný systém omnoho jednoduchší a intuitívnejší. Ďalším mínusom je príliš veľký počet výrobných a skladových sektorov, čo vedie k neprehľadnosti pohybu materiálu a taktiež k neprehľadnosti rozpracovanosti výroby.

Postupnou implementáciou opravných patchov sa systém aj po roku stále vylepšuje podľa požiadaviek zákazníka, avšak stále je to beh na dlhé trate.

Kapitola 3

Špecifikácia a návrh systému

Návrh systému a jeho špecifikácia vychádza z vykonanej analýzy. Pozostáva zo špecifikácie používateľov systému, ktorá je postavená na informáciách o používateľoch zo spomínanej analýzy, a z návrhu štruktúry databázy, ktorá vychádza z požadovanej funkčnosti a zistených informácií z reálneho prostredia výrobnej firmy.

3.1 Používatelia systému

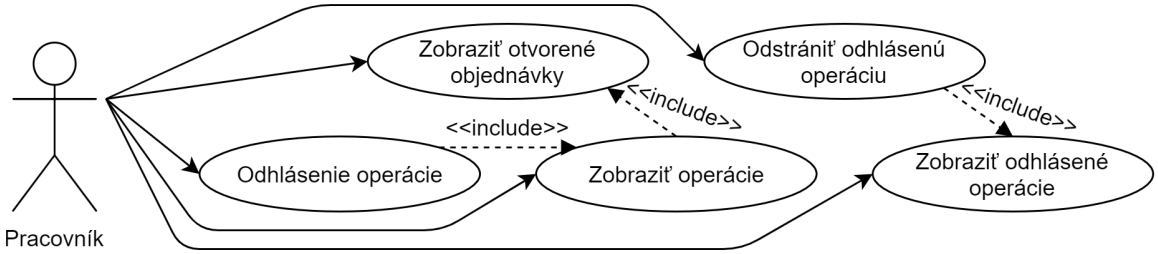
Ako vyplýva z analýzy výrobného procesu opísanej v kapitole 2.1, v systéme sa bude vyskytovať 6 rôznych rolí používateľov, a to:

- Pracovník
- Skladník
- Ekonóm
- Technik
- Vedúci
- Administrátor

Každej z týchto rolí sa teraz budeme venovať podrobnejšie. Celý návrh systému v podobe diagramu prípadu použitia môžeme vidieť v prílohe C.

3.1.1 Pracovník

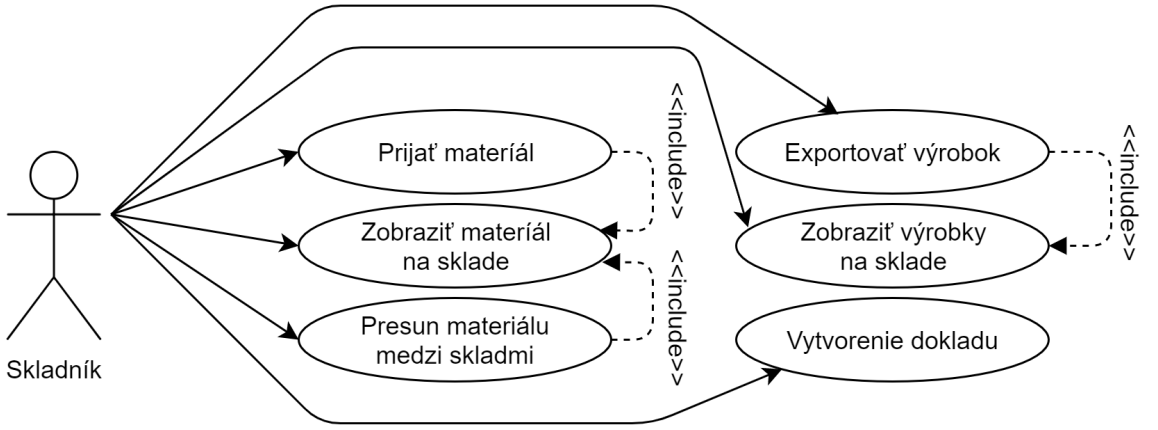
Rola pracovníka je najjednoduchšou základnou rolou systému. Pracovník má prístup len do výrobnej časti systému a môže si zobraziť zoznam orvorených objednávok a po otvorení konkrétnej objednávky zo systému môže postupne odhlasovať jednotlivé operácie potrebné na výrobu daného produktu. Taktiež môže vydieť zoznam všetkých svojich odhlásených operácií a v prípade, že omylom odhlásil nesprávnu operáciu alebo odhlásil v operácií zlé množstvo, môže túto svoju odhlásenú operáciu odstrániť a tá bude znovu dostupná v zozname operácií.



Obr. 3.1: Diagram prípadov použitia pre rolu pracovníka

3.1.2 Skladník

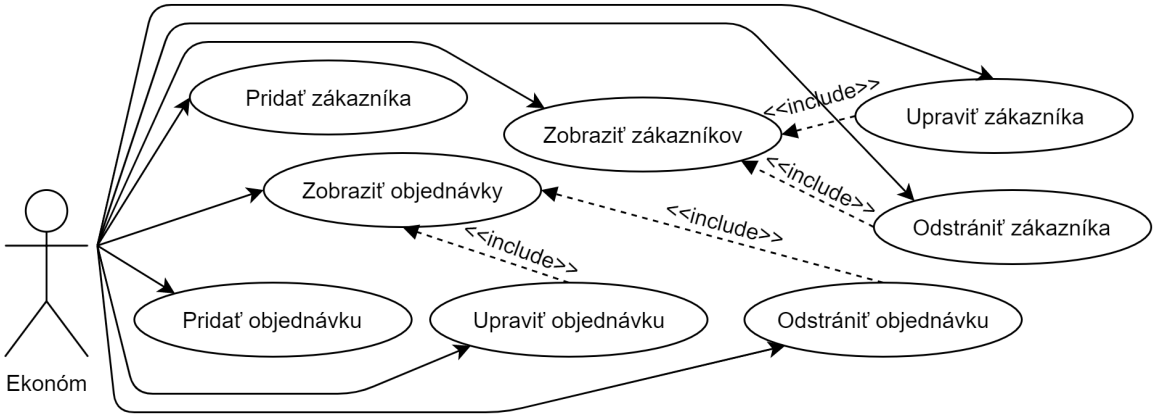
Rola skladníka sa bude starať hlavne o materiál a hotové výrobky na sklade. Bude môcť prijímať nový materiál na sklad, zobraziť si zoznam všetkých materiálov na všetkých skladoch a taktiež presúvať materiál medzi jednotlivými skladmi. Čo sa týka hotových výrobkov, bude si môcť zobraziť ich zoznam na skladoch a exportovať ich zo skladu. Pri exporte bude musiť skladník vytlačiť sprievodný doklad ku každému výrobku, ktorý bude exportovať. Pri tlači si vyberie výrobok, ktorý ide exportovať a taktiež počet kusov.



Obr. 3.2: Diagram prípadov použitia pre rolu skladníka

3.1.3 Ekonóm

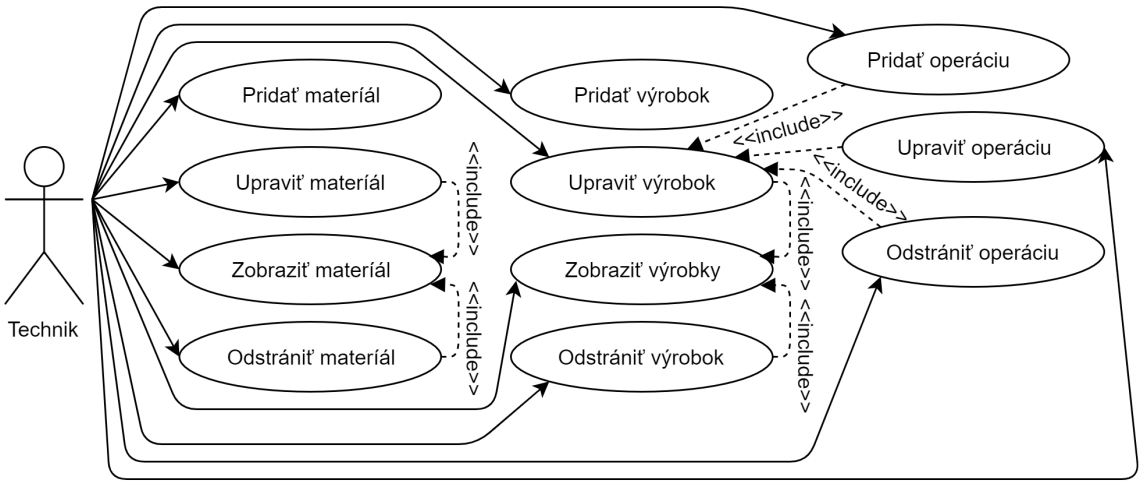
Ekonóm má na starosti finančnú stránku informačného systému. Môže teda prijať novú objednávku od zákazníka a zadať ju do systému. Taktiež po exportovaní všetkých kusov objednaného produktu môže ekonóm uzavrieť danú objednávku. Ekonóm tiež môže do systému pridať nového zákazníka alebo upraviť existujúceho.



Obr. 3.3: Diagram prípadov použitia pre rolu ekonóma

3.1.4 Technik

Technik alebo technológ je jedna z najdôležitejších rolí v systéme. Má nastarosti pridávanie a úpravu nových produktov, poprípade nového materiálu do systému. Tvorba nového materiálu je jednoduchá, je potrebné zadať len názov a popis materiálu. Tvorba nového produktu je náročnejšia. Najskôr je potrebné zadať názov, popis a cenu nového produktu. Potom je nutné vybrať zákazníka, pre ktorého bude tento produkt určený a treba vybrať sklad, na ktorý sa budú ukladať hotové výrobky. Nakoniec treba ku produktu pridať operácie, ktoré je nutné vykonať pre výrobu tohto produktu. Operácie sú popísané v kapitole 3.2.6.

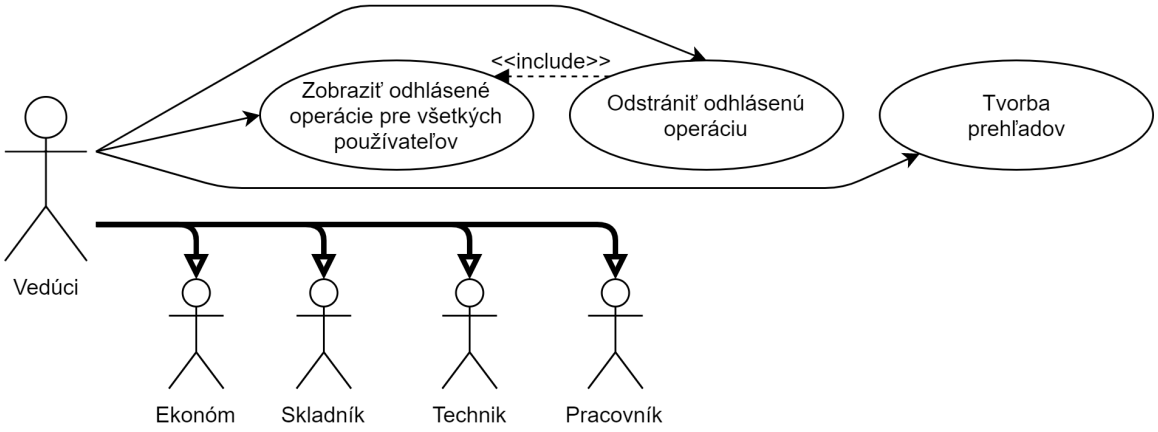


Obr. 3.4: Diagram prípadov použitia pre rolu technika

3.1.5 Vedúci

Rola vedúceho bude mať prístup ku všetkým funkciám systému, ako role pracovníka, skladníka, ekonóma a technika. Navyše môže vedúci vytvárať prehľady, ktoré budú exportované vo formáta xlsx. Vedúci môže vytvoriť prehľad výkonov jednotlivých zamestnancov alebo prehľad objednávok za určité časové obdobie. Vedúci tiež môže vydieť všetky odhlásené

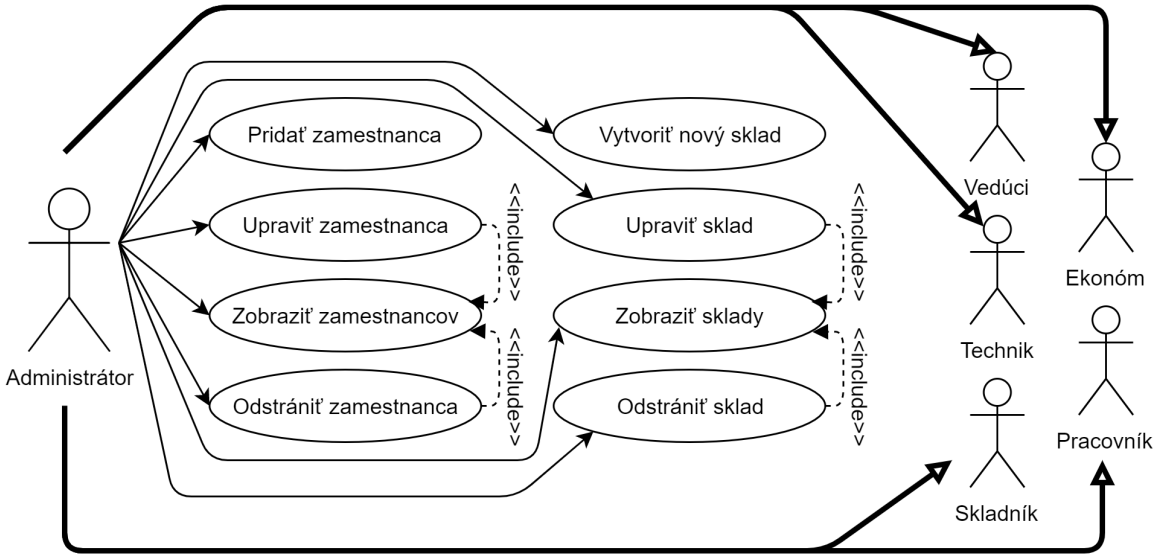
operácie pre všetkých používateľov systému a v prípade potreby ich odstrániť, ak nastala chyba pri odhlasovaní.



Obr. 3.5: Diagram prípadov použitia pre rolu vedúceho

3.1.6 Administrátor

Administrátor je hierarchicky najvyššia rola v systéme. Má prístup do všetkých častí systému a oproti ostatným používateľom môže v systéme vykonávať najdôležitejšie zmeny, ako pridávať, upravovať alebo odstraňovať sklady, ale aj samotných používateľov. Kedže sa jedná o uzavretý systém, nový používateľ sa nemôže voľne zaregistrovať, pridať zo systému ho môže práve administrátor. Ten mu tiež pridelí niektoré role systému.



Obr. 3.6: Diagram prípadov použitia pre rolu administrátora

3.2 Databáza

Jednou z najdôležitejších častí informačného systému je jeho databázová vrstva, alebo databáza. Preto sme tejto časti venovali obzvlášť pozornosť. Celý návrh databázy je možné vidieť v príloženom ER Diagrame v prílohe B. V najbližších sekciách si popíšeme jednotlivé tabuľky relačnej databázy, ktoré sú generované automaticky zo špecifikovaných entít pomocou O-R mapovania. Viac o entitách je popísané v kapitole 5.2.

3.2.1 Authorities

username	authority

Tabuľka 3.1: Tabuľka **authorities**

Tabuľka **authorities** slúži na ukladanie priradených rolí jednotlivým užívateľom. Kedže jeden používateľ systému môže mať priradených viac ako jednu rolu, je potrebná takáto tabuľka, ktorá prepája ďalšie dve tabuľky `users` a `roles`. Obsahuje zložený primárny kľúč z oboch stĺpcov tabuľky, teda `username` a `authority`. Tabuľka sa využíva pri kontrole prístupu k jednotlivým častiam systému.

3.2.2 Customers

id	name

Tabuľka 3.2: Tabuľka **customers**

Tabuľka **customers** obsahuje jednotlivých zákazníkov, pre ktorých sa bude vyrábať nejaký produkt. Primarnym kľúčom je stĺpec `id`, ktorý je automaticky generovaný. Ďalej obsahuje meno zákazníka `name`. Využíva sa pri definovaní nového produktu, alebo aj pri vkladaní objednávok do systému.

3.2.3 Documents

id	date_time	quantity	order_id

Tabuľka 3.3: Tabuľka **documents**

Do tabuľky **documents** sa ukladajú všetky sprievodné dokumenty, ktoré sa generujú pri exportovaní hotových produktov skladníkom. Využíva sa pri generovaní sprievodného dokumentu vo formáte PDF, ktorý je potrebný pre export produktu. Primárnym kľúčom je automaticky generovaný stĺpec `id`, ďalej obsahuje stĺpec `date_time`, ktorý určuje čas prvého generovania PDF dokumentu, stĺpec `quantity` určujúci počet hotových produktov na export a posledným stĺpcom je `order_id`, čo je cudzí kľúč odkazujúci na tabuľku `orders`.

3.2.4 Done_operations

id	username	order_id	operation_id	quantity	date_time

Tabuľka 3.4: Tabuľka **done_operations**

Tabuľka **done_operations** slúži na uchovanie histórie vykonaných operácií pre jednotlivých užívateľov. Taktiež môže slúžiť pri tvorbe prehľadov. Primarným kľúčom je automaticky generovaný identifikátor `id`, ďalej obsahuje užívateľské meno `username` používateľa, čo je cudzí kľúč do tabuľky `users`, cudzí kľúč `order_id` identifikátor objednávky, cudzí kľúč `operation_id` identifikátor operácie, `quantity` určujúce odhlásené množstvo a `date_time` je dátum a čas odhlásenia operácie.

3.2.5 Materials

id	name	description

Tabuľka 3.5: Tabuľka **materials**

V tejto tabuľke sú úložené informácie o jednotlivých materiáloch, ktoré sa používajú na výrobu. `id` je automaticky generovaný primárny kľúč. Ďalej obsahuje stĺpec `name`, v ktorom je uložený názov materiálu a stĺpec `descritpion`, kde je uložený popis materiálu.

3.2.6 Operations

id	product_id	name	time	order_number	description	storage_id	dest_storage_id

Tabuľka 3.6: Tabuľka **operations**

Tabuľka **operations** slúži na definovanie operácií potrebných k výrobe daného produktu. Každá operácia je naviazaná presne na jeden produkt. Tabuľka sa využíva hlavne pri odhlasovaní výroby. Primárnym kľúčom je automaticky generovaný identifikátor `id`, cudzí kľúč `product_id` do tabuľky `products`, stĺpec `name` definujúci názov operácie, `time` určujúci čas potrebný na splnenie tejto operácie pre 1 kus, `order_number` určujúci poradie operácií, `description` určujúci popis operácie, cudzí kľúč `storage_id` odkazujúci na tabuľku `storages`, ktorý určuje sklad, z ktorého sa majú brať materiály potrebné pre splnenie operácie a posledným stĺpcom je cudzí kľúč `dest_storage_id` taktiež odkazujúci na `storages`, ktorý určuje sklad, na ktorý sa má ukladať hotový výrobok.

3.2.7 Operations_materials

operation_id	material_id	quantity

Tabuľka 3.7: Tabuľka **operations_materials**

Tabuľka **operations_materials** slúži ako pomocná tabuľka pre ukladanie materiálu a jeho počtu potrebného pre vykonanie konkrétnej operácie. Využíva zložený primárny kľúč z dvoch stĺpcov tabuľky, a to `operation_id` a `material_id`. Ďalej obsahuje stĺpec `quantity`, ktorý určuje množstvo materiálu potrebného pre vykonanie operácie.

3.2.8 Orders

id	customer_order_id	product_id	quantity	date	price	open

Tabuľka 3.8: Tabuľka **orders**

Do tabuľky **orders** sa ukladajú novo prijaté objednávky do systému. Jedna objednávka môže byť len na jeden typ produktu ale v ľubovoľnom množstve. Primárnym kľúčom tabuľky je `id`, ktoré je automaticky generované. Ďalej obsahuje stĺpec `customer_order_id`, v ktorom je uložený identifikátor objednávky zo strany zákazníka, stĺpec `product_id`, ktorý je cudzí kľuč do tabuľky `products` a určuje produkt, ktorý je objednaný, ďalej stĺpec `quantity`, ktorý určuje objednané množstvo, stĺpec `date`, ktorý určuje termín dodania, stĺpec `price`, kde je uložená cena celej objednávky a posledným stĺpcom je príznak `open` ktorý určuje, či je daná objednávka otvorená alebo uzavretá.

3.2.9 Orders_operations

order_id	operation_id	quantity_done

Tabuľka 3.9: Tabuľka **orders_operations**

Order_operations je pomocná tabuľka pre priraďovanie operácií konkrétneho výrobku ku niektorej z operácií. Keďže niektorý pordukt môže byť objednaný niekoľkokrát, pre každú objednávku si musíme pamätať samostatný zoznam potrebných operácií. Tabuľka obsahuje zložený primárny kľúč zložený zo stĺpcov `order_id` a `operation_id`. Ďalej obsahuje stĺpec `quantity_done`, v ktorom je uložené dokončené mnôžstvo pre každú operáciu.

3.2.10 Products

id	name	customer_id	price	description

Tabuľka 3.10: Tabuľka **products**

Do tabuľky **products** sú uložené všetky produkty v informačnom systéme. Práve na túto tabuľku je naviazaná tabuľka `operations` a jednotlivé operácie v nej. Primárnym kľučom tejto tabuľky je automaticky generovaný identifikátor `id`, ďalej obsahuje názov produktu `name`, cudzí kľúč `customer_id` do tabuľky `customers` určujúci zákazníka, pre ktorého je daný produkt určený, ďalej `price`, teda cenu produktu a posledným je `description` popis produktu.

3.2.11 Roles

name

Tabuľka 3.11: Tabuľka **roles**

Tabuľka **roles** je veľmi jednoduchá. Využíva sa pri vytváraní nového používateľa a pri úprave existujúcich používateľov systému. Jej primárnym kľúčom a zároveň aj jediným stĺpcom je `name`, určujúci názov roly.

3.2.12 Storages

id	name

Tabuľka 3.12: Tabuľka **storages**

V tabuľke **storages** sú uložené rôzne sklady nachádzajúce sa v systéme. Využíva sa pri práci s materiálom ale aj hotovými výrobkami, ktoré sú na určitom sklade uložené. Primárnym kľučom tabuľky je stĺpec `id` , ktorý je automaticky generovaný. Ďalej tabuľka obsahuje už len stĺpec `name` určujúci názov skladu.

3.2.13 Storages_materials

storage_id	material_id	quantity

Tabuľka 3.13: Tabuľka **storages_materials**

Táto tabuľka slúži ako pomocná tabuľka na uloženie počtu určitého materiálu na niektorom zo skladov. Keďže jeden druh materiálu sa môže nachádzať na viacerých skladoch súčasne, je potrebná práve takáto tabuľka na uchovanie tejto informácie. Primárny kľuč je zložený a pozostáva zo stĺpcov `storage_id` a `material_id`. Tabuľka ešte obsahuje stĺpec `quantity`, ktorý určuje počet materiálu s identifikátor `material_id` na sklade s identifikátor `storage_id`.

3.2.14 Storages_products

storage_id	product_id	quantity

Tabuľka 3.14: Tabuľka **storages_products**

Storages_products je ďalšia pomocná tabuľka. Táto slúži na uchovanie informácie o počte hotových výrobkov na skladoch. Je veľmi podobná tabuľke `storages_materials`, rovnako obsahuej zložený primárny kľúč `storage_id` a `product_id` a tiež obsahuje stĺpec `quantity` určujúci počet hotových výrobkov s identifikátorom `product_id` na sklade s identifikátorom `storage_id`.

3.2.15 Users

first_name	last_name	username	password	enabled

Tabuľka 3.15: Tabuľka **users**

Poslednou tabuľkou v databáze je tabuľka **users**, v ktorej sú uložené informácie o všetkých používateľoch systému. Patrí medzi najdôležitejšie tabuľky, keďže sa využíva hlavne pri prihlasovaní užívateľa do systému. Primárnym kľúčom je uživateľské meno `username`, ďalej obsahuje meno `first_name` a priezvisko `last_name`, ďalej obsahuje heslo `password`, ktoré je v databáze uložené v bezpečnej forme, keďže je zašifrované pomocou algoritmu BCrypt,

a posledným stĺpcom v tabuľke je stĺpec `enabled`, kde je uložený príznak, či je daný užívateľ povolený pre systém alebo nie.

Kapitola 4

Použité technológie

Keďže predmetom návrhu je informačný systém na webovom rozhraní, tomu je prispôsobený aj výber použítých technológií. V nasledujúcich sekciách sú tieto technológie popísané.

4.1 HTML

HTML (Hypertext Markup Language) je jazyk používaný na tvorbu webových stránok. Je to základný stavebný kameň každého webu. „Hypertext“ odkazuje na hypertextové prepojenia, ktoré môžu jednotlivé HTML stránky obsahovať. Práve tieto prepojenia sú základom fungovania dnešného internetu. „Markup Language“ odkazuje na značky, ktoré sa používajú na definovanie jednotlivých prvkov HTML stránky.[7]

HTML definuje jednak význam, ale aj štruktúru každej webovej stránky. Veľmi často sa spoju s jazykom HTML používajú ďalšie technológie, ako napríklad CSS, slúžiace na definovanie vizuálnej časti stránky, alebo JavaScript, ktorý sa stará o funkcionalitu a správanie stránky. Týmto technológiám sa podrobnejšie venujeme v ďalších podkapitolách.

Dokument vo formáte HTML sa skladá zo značiek (tagov), ktoré pozostávajú z názvu elementu obklopeného znakmi „<“ a „>“. Názov elementu je *case insensitive*, teda nezáleží na veľkosti jednotlivých písmen.[7]

Každý HTML dokument by mal začínať tagom `<!DOCTYPE html>` čím definujeme, že nasledujúci dokument je napísaný práve v jazyku HTML. Zvyšok HTML dokumentu po definovaní jeho typu musí byť obklopený tagmi `<html>` a `</html>`. Táto časť dokumentu sa dá rozdeliť na dve hlavné časti, hlavičku a telo dokumentu. Hlavička je v jazyku HTML ohraničená párovým tagom `<head></head>`. Na rozdiel od elementov v tele dokumentu, elementy v hlavičke nie sú viditeľne na stránke po načítaní v prehliadači. Hlavička slúži na definovanie informácií o dokumente, napríklad nadpisu, autora, znakovej sady, štýlov a taktiež na importovanie rôznych súborov s kaskádovými štýlmi alebo súborov so skriptmi. Druhá časť dokumentu, telo, je ohraničené taktiež párovým tagom `<body></body>`. Ako sme už spomenuli, prvky v tele dokumentu sú viditeľné po načítaní stránky v prehliadači. Obsahuje teda obsah stránky ako text, obrázky, prepojenia, tabuľky atď.[26]

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Titulok stránky</title>
    </head>
    <body>
        <h1>Nadpis prvej úrovne</h1>
        <p>Ukážka jednoduchého HTML dokumentu.</p>
    </body>
</html>
```

Obr. 4.1: Príklad HTML dokumentu

4.2 CSS

CSS (Cascading Style Sheets) je jazyk používaný na definovanie vizuálneho charakteru webovej stránky. Názov v preklade znamená kaskádové štýly, pretože sa na seba môžu vrstviť viaceré definície štýlov. Jazyk CSS definuje vzhľad dokumentu nezávisle od jeho obsahu. To má viacero výhod, napr. je možné definovať jeden vzhľad pre viacero HTML dokumentov, čo zaručí konzistentný vzhľad stránok alebo jednotný spôsob definície štýlu pre všetky elementy.[5]

Syntax jazyka CSS je pomerne jednoduchá. Skladá sa z dvoch hlavných častí, selektoru a deklarácie vlastností. Tiež je možné určiť viacero selektorov pre jedno pravidlo alebo viacero pravidiel pre rovnaký selektor. Selektor určuje, pre ktoré elementy bude dané pravidlo platné. Poznáme viacero druhov selektorov, ako prázdny selektor (alebo tiež *), kde pravidlo platí pre všetky elementy, meno elementu, kde pravidlo platí pre všetky elementy daného mena, identifikátor elementu (identifikátor musí predchádzať znak „#“), kde pravidlo platí pre elementy s danou hodnotou atribútu id a trieda elementu (triedu musí predchádzať znak „.“), kde pravidlo platí pre všetky elementy s danou hodnotou atribútu `class`. Deklarácia vlastností má tvar `vlastnosť: hodnota;` kde každá vlastnosť má svoje meno, napr. `display, border` a taktiež má každá vlastnosť množinu prípustných hodnôt, napr. vlastnosť `width` môže obsahovať číselnú hodnotu nasledovanú jednotkami `px, em, rem, %` a taktiež môže obsahovať hodnotu `auto`.[25]

```css
h1{ color: blue; }

#identif{
    border: 2px solid red;
    font-size: 32px;
}

.trieda .ina_trieda{
    text-align: center;
    margin: 10px 20px 30px 40px;
}
```

Obr. 4.2: Príklad kaskádových štýlov CSS

4.3 JavaScript

JavaScript je ako už z názvu vychádza skriptovací programovací jazyk. Používa sa pri tvorbe webových stránok na vloženie funkcionality do danej stránky. Je to interpretovaný jazyk, čo znamená, že sa nemusí prekladať. Rovnako je to jazyk objektový, teda program sa skladá z viacerých častí, objektov, z ktorých každý má na zodpovednosť inú funkcionalitu, a tiež je *case sensitive*, takže záleží na veľkosti písmen. Program v JavaScript-e sa zo serveru, kde je zvyčajne uložený spolu s webovou stránkou ktorú sa pokúša klient načítať, odosiela do prehliadača klienta a až tam sa interpretuje. [8]

Jazyk JavaScript je slabo typovaný a dynamicky typovaný programovací jazyk. Slabo typovaný znamená, že môžeme robiť operácie aj s rôznymi dátovými typmi a jazyk si sám urobí internú konverziu medzi týmito typmi, aby mohol operáciu vykonať. Dynamicky typovaný znamená, že pri deklarovaní premennej neurčujeme jej typ. Ten sa nastaví až pri priradení prvej hodnoty do tejto premennej.[18] JavaScript obsahuje 6 dátových typov a to number (číselné hodnoty), string (reťazce), boolean (logická hodnota), null, undefined (v tomto stave sa nachádza premenná pri jej deklarovaní bez priradenia hodnoty) a object.[28]

Kód v jazyku JavaScript môžeme rozdeliť do dvoch hlavných zložiek a to samotný jazyk a aplikačné rozhranie, s ktorým jazyk spolupracuje. Práve toto rozhranie umožňuje samotnému jazyku spolupracovať sa daným prehliadačom vďaka svojim objektom a ich metódam.[8]

JavaScript je jedným z najpoužívanejších programovacích jazykov na svete.[19] Využíva sa na tvorbu dynamického obsahu webovej stránky, napríklad generovanie obsahu stránky, vkladanie aktuálnych informácií (čas, dátum atď.), ďalej sa používa na zachytávanie rôznych udalostí, ako kliknutie myši, stlačenie klávesnice atď. a v súčasnosti je veľmi populárna funkcia tohto jazyka načítanie častí stránky bez jej celého obnovenia, čo sa označuje ako AJAX (Asynchronous JavaScript and XML).

```
1  function ChangeContent(id){
2      var element = document.getElementById(id);
3      element.innerHTML = "Obsah elementu sa zmenil";
4  }
```

Obr. 4.3: Príklad funkcie v jazyku JavaScript

4.4 JQuery

JQuery je knižnica jazyka JavaScript navrhnutá pre uľahčenie používania jazyka JavaScript. JQuery zoberie veľké množstvo riadkov v JavaScript-e, ktoré sú potrebné na vykonanie danej funkcie a zabalí ich do metódy, ktorá nezaberá viac ako jeden riadok kódu. To prispieva k sprehľadneniu zdrojového kódu a urýchli prácu programátora. Táto knižnica obsahuje vlastnosti na manipuláciu s HTML/DOM, úpravu CSS štýlov, odchytávanie udalostí, efekty, animácie atď.[29]

Základná syntax knižnice JQuery je veľmi jednoduchá. Skladá sa z dvoch častí, selektoru a akcie, ktorú chceme nad vybraným elementom uskutočniť a vyzerá nasledovne: `$(selector).action()`. To, že chceme využiť niektorú z metód knižnice JQuery dáme najavo použitím znaku „$“, knižnica následne pomocou selektoru vyberie jeden alebo viacero HTML prvkov a aplikuje na ne špecifikovanú akciu, ako napríklad zmenu CSS štýlu. JQuery používa pre selektory syntax jazyka CSS, teda môže vybrať elementy určitého názvu

(`$("h1").fadeIn()`), ďalej môže vybrať element s daným atribútom id (`$("#identif").fadeIn()`) a tiež môže vybrať všetky elementy danej triedy (`$(".trieda").fadeIn()`).[30]

```
1   $("#tlacitko").click(function(){
2       $(".trieda").show();
3       $("p").css("color", "blue");
4   });
```

Obr. 4.4: Príklad využitia knižnice JQuery

4.5 Bootstrap

Bootstrap je sada nástrojov a šablón pre tvorbu webových stránok a aplikácií. Tieto šablóny sú založené na HTML, CSS a JavaScript-e a slúžia na úpravu elementov v základnom HTML ako sú formuláre, tlačidlá a podobne. Pre použitie Bootstrap-u a jeho JavaScript-ovej časti je potrebné JQuery, ktoré je opísané v predchádzajúcej kapitole.[24] Bootstrap je voľne stiahnuteľný a jeho aktuálna verzia je 4.4.1 (november 2019).[2] V súčasnosti sa najviac používa pre tvorbu responzívnych webových stránok.

Tento framework má viacero výhod, medzi ktoré patria napríklad jednoduchosť, takže na použitie Bootstrap-u stačí ovládať základy HTML a CSS, ďalej vlastnosti na tvorbu responzívneho designu, ktorý je v súčasnosti veľmi dôležitý, keďže používatelia prehliadajú webové stránky na rôznych zariadeniach, a neposlednou výhodou je kompatibilita so všetkými aktuálne používanými prehliadačmi, takže pri jeho použití sa nemusíme obávať problémov pri používaní rôznych prehliadačov.[24]

4.6 SQL

SQL (Structured Query Language) je programovací jazyk používaný na komunikáciu s databázami. V súčasnosti je SQL najpoužívanejší jazyk pre prácu s relačnými databázami.[19] Aj keď je tento jazyk štandardizovaný inštitúciou ANSI (American National Standards Institute), existuje viacero jeho verzií, všetky však podporujú väčšinu najpoužívanejších príkazov. [27]

Ako sme už spomenuli, jazyk SQL je používaný v relačných systémoch riadenia databáz. Existuje viacero aktuálne používaných databázových systémov, medzi najznámejšie patria Microsoft SQL Server, Oracle alebo MySQL. Každý z týchto systémov pridáva do čistého jazyka nejaké rozšírenia, väčšinou na doplnenie funkcií procedurálneho programovacieho jazyka. Systému MySQL sa budeme podrobnejšie venovať neskôr v tejto kapitole. [15]

Jazyk SQL sa používa nielen na vyhľadávanie údajov v databázach, ale aj na veľa ďalších operácií, ako vytváranie tabuliek, vkladanie údajov do tabuliek, modifikáciu dát, vymazanie tabuľky, prideľovanie práv atď. Príkazy jazyka SQL sa dajú rozdeliť na štyri základné kategórie na základe ich využitia[1]:

- jazyk pre definovanie dát
- jazyk pre manipuláciu s dátami
- jazyk pre kontrolu transakcií
- jazyk pre kontrolu dát

Prvá kategória je jazyk pre definovanie dát(*Data Definition Language (DDL)*). Príkazy z tejto katégorie slúžia na vytváranie, modifikáciu a vymazávanie databázových objektov. Medzi tieto príkazy patria `CREATE`, `ALTER`, `DROP`, `RENAME` a `TRUNCATE`. `CREATE` vytvára nové databázové objekty, ako samotné databázy, tabuľky, funkcie atď., `ALTER` sa používa na úpravu databázových objektov, príkazom `DROP` odstránime vybrané databázové objekty, `RENAME` na zmenu názvu databázového objektu a `TRUNCATE` na vymazanie dát z databázového objetku, samotný databázový objekt však ostane v systéme.[1]

Druhá kategória je jazyk pre manipuláciu s dátami (*Data Manipulation Language (DML)*). Tieto príkazy slúžia pre ukladanie, modifikáciu, vymazávanie a získavanie dát z databázy. Patria sem príkazy `SELECT` na získanie špecifických dát z databázy, `INSERT` na vkladanie dát do databázy, `UPDATE` na aktualizovanie dát v databátozovom objekte a `DELETE` na odstránenie dát z databázy.[1]

Ďalšou kategóriou je jazyk pre kontrolu transakcií (*Transaction Control Language (TCL)*). Príkazy tejto kategórie slúžia na vykonávanie zmien ovplyvňujúcich dáta. Patria sem `COMMIT`, `ROLLBACK` a `SAVEPOINT`. `COMMIT` slúži na zapísanie vykonaných zmien do databázového objektu, `ROLLBACK` na vrátenie vykonaných zmien ku poslednej transakcií alebo ku poslednému záchytnému bodu a `SAVEPOINT` na vytváranie záchytných bodov.[1]

Posledná kategória je jazyk pre kontrolu dát (*Data Control Language (DCL)*). Príkazy z tejto kategórie slúžia na zabezpečenie dát v databázových objektoch. Patria sem `GRANT` na pridelenie práv používateľom ku prístupu k databázovým objektom a `REVOKE`, ktorý tieto práva odoberá.[1]

4.6.1 MySQL

MySQL je jeden z prvých open-source relačných systémov riadenie databáz. Tento systém je podporovaný na takmer všetkých platformách (napr. Windows, Linux, macOS) a taktiež je implementovaný vo veľkej väčšine používaných programovacích jazykov, ako napríklad C++, PHP alebo Java. MySQL je postavené na modely klient-server. Tento systém je vhodný pre malé aj veľké projekty, je veľmi rýchly, spoľahlivý a jednoduchý na použitie. V súčasnosti je vyvíjaný, distribuovaný a podporovaný spoločnosťou Oracle Corporation. [31]

Každá databáza v MySQL je tvorená jednou alebo viacerými tabuľkami, kde každá tabuľka pozostáva z riadkov a stĺpcov.[31] Riadok je jeden záznam v databáze a stĺpec reprezentuje jednotlivú položku informácie – typ položky, ktorý sa nachádza v každom zázname. Tieto informácie môžu byť viacerých dátových typov. Tieto dátové typy sa delia na reťazcové (napr. `CHAR`, `VARCHAR`, `TEXT`), číselné (napr. `BIT`, `INT`, `FLOAT`) a dátumové a časové (napr. `DATETIME`, `DATE`, `TIMESTAMP`) dátové typy.[9] S MySQL sa pracuje pomocou takzvaných dotazov. Tieto vychádzajú z jazyka SQL, môžu však byť rozšírené o funkcionalitu, ktorú MySQL pridáva.

```sql
--VYTVORÍME DATABÁZU
CREATE DATABASE `skola`;

--VYBERIEME DATABÁZU NA POUŽÍVANIE
USE `skola`;

--VYTVORÍME TABUĽKU
CREATE TABLE `studenti`(
    `id` INT(11) auto_increment,
    `meno` VARCHAR(32),
    `priezvisko` VARCHAR(32),

    PRIMARY KEY(`id`)
)

--VLOŽÍME ZÁZNAM DO TABUĽKY
INSERT INTO studenti(meno, priezvisko) VALUES(`Ján`, `Novák`);

-- VYBERIEME VŠETKY ZÁZNAMY TABUĽKY
SELECT * FROM studenti;
```

Obr. 4.5: Príklad použitia jazyka SQL

4.7 Java

Java patrí medzi najpoužívanejšie programovacie jazyky a aktuálne je to najpopulárnejší programovací jazyk na svete (Marec 2020). [19] Pri tvorbe Javy sa vývojári držali piatich základných princípov, a to že Java musí byť[17]:

- Jednoduchá, objektovo orientovaná a povedomá
- Robustná a bezpečná
- Nezávislá na architektúre a prenositeľná
- Výkonná
- Interpretovaná, viacvláknová a dynamická

Java je univerzálny, objektovo orientovaný, triedne založený programovací jazyk, ktorý dokáže využívať viaceré vlákna pre paralelné spracovanie operácií. Dôležitou súčasťou a taktiež výhodou jazyka Java je virtuálny stroj Javy, na ktorom preložené programy v tomto jazyku bežia, teda je to jazyk interpretovaný. Jednoduchosť tohto jazyka spočíva v tom, že odstránil zložité súčasti ako ukazovatele a iné, ktoré môžeme vydieť v iných podobných jazykoch ako napríklad C++. Objektovo-orientovaný jazyk sa na všetko pozerá ako na objekt a aj všetky operácie sú vykonávané s využitím týchto objektov. Ďalšou výhodou Javy je jej bezpečnosť, keďže všetko preložený kód je prekonvertovaný do tzv. bytekódu, ktorý nie je jednoducho čitateľný. Robustnosť tkvie v silnom systéme na riadenie pamäti, čo taktiež pomáha pri včasnom odhaľovaní chýb. Taktiež to znamená, že programátor sa nemusí starať o uvoľňovanie pamäti, to za neho rieši tzv. *garbage collector*, ktorý automaticky uvoľní pamäť, keď nezostanú žiadne odkazy na určitý objekt. Výkonnosť tohto jazyka zabezpečuje už spomínaný bytekód, ktorý je ľahko preložiteľný do strojového kódu. Výkonnosti taktiež

napomáha *JIT (Just In Time)* prekladač, teda preklad prebieha za behu. Neposlednou výhodou Javy je jej multiplatformnosť, teda je nezávislá na platforme.[6]

4.7.1 Komponenty Javy

Java pozostáva z troch základných komponentov. Virtuálny stroj Javy (*Java Virtual Machine (JVM)*), *Java Runtime Environment (JRE)* a *Java Development Kit (JDK)*.[6]

Virtuálny stroj Javy je, ako už názov napovedá, abstraktný stroj. Je to špecifikácia, ktorá poskytuje prostredie pre prostredie, v ktorom môže bytekód Javy bežať. Pozostáva zo špecifikácie, ktorá opisuje implementáciu virtuálneho stroja Javy, ďalej samotnej implementácie a nakoniec preloženej behovej inštancie, ktorá bude spustená pri zavolaní virtuálneho stroja Javy.[6]

Java Runtime Environment odkazuje na behové prostredie, v ktorom môže byť preložený bytekód spustený. To znamená že JRE implementuje virtuálny stroj Javy a navyše poskytuje všetky potrebné triedy a ďalšie potrebné súbory, ktoré JVM potrebuje pre svoj beh.[6]

Posledným komponentom je Java Development Kit. Je to nástroj, ktorý je potrebný pre preklad, dokumentáciu a balenie programov napísaných v jazyku Java. JDK obsahuje všetky potrebné triedy a súbory ako JRE, navyše však poskytuje interpreter, prekladač (*javac*), nástroj na balenie programov (*jar*) a generátor dokumentácie (*javadoc*).[6]

4.7.2 Java platformy

V súčasnej dobe existujú štyri platformy Javy:[16]

- Java SE
- Java EE
- Java ME
- JavaFX

Obr. 4.6: Java platformy[16]

Java SE (Standard Edition) poskytuje základnú funkcionalitu jazyka. Definuje všetko od základných dátových typov a objektov až po zložité triedy využívané pre budovanie sietí, prístupy do databázy, tvorbu grafických užívateľských rozhraní atď.[16]

Java EE (Enterprise Edition) je platforma jazyka Java využívaná pre tvorbu podnikových aplikácií a informačných systémov. Tejto platforme sa podrobnejšie budeme venovať v nasledujúcej podkapitole.[16]

Java ME (Micro Edition) poskytuje mobilné rozhranie pre programovanie aplikácií (API) a zmenšenú verziu virtuálneho stroja Javy (Java Virtual Machine) aby aplikácie takto tvorené mohli byť spustené na menších zariadeniach, ako mobilné telefóny atď. Toto rozhranie obsahuje niektorú funkčnosť platformy Java SE a navyše zahŕňa špeciálne triedy, ktoré napomáhajú tvorbe aplikácií pre tieto menšie zariadenia.[16]

JavaFX je platforma, ktorá napomáha jednoduchšej tvorbe klientskych aplikácií. Aplikácie vytvorené na tejto platforme využívajú hardvérovú akceleráciu aby vo väčšej miere využili výhody výkonnejších klientov.[16]

4.7.3 Java Enterprise Edition

Java EE je, ako už z názvu vyplýva, určená pre podniky a väčšie firmy. Je postavená na štandardnej edícií Javy (Java SE), rozširuje ju však o množstvo štandardov, knižníc a modulov vhodných pre vývoj podnikových aplikácií.[10] Podnikové aplikácie by sa dali charakterizovať nasledujúcimi bodmi:

- Obsluhujú naraz veľké množstvo užívateľov
- Pracujú s veľkým množstvom dát v databázach
- Komunikujú s ďalšími systémami
- Sú robustné a bezpečné

Podniková verzia Javy sa teda snaží poskytnúť čo najviac prostriedkov na vytváranie takýchto aplikácií. Je založená na takzvanej trojvrstvej architektúre, v ktorej je aplikácia rozdelená na databázovú vrstvu, vrstvu obchodnej logiky a prezentačnú vrstvu. [14]

```
1  class TestClass{
2      public static void main(String[] args) {
3          System.out.println("Hello world!");
4      }
5  }
```

Obr. 4.7: Príklad jednoduchej triedy v jazyku Java

4.8 Spring Framework

Spring Framework je momentálne najpopulárnejší aplikačný rámec pre vývoj podnikových aplikácií na platforme Java EE. Tento rámec vo veľkej miere uľahčuje a urýchľuje prácu programátora a taktiež za neho rieši veľa otázok týkajúcich sa napríklad bezpečnosti. Základné vlastnosti Spring Frameworku môžu byť použité pre vývoj Java aplikácií, ale vďaka množstvu nadstavieb je však omnoho užitočnejší pri tvorbe webových aplikácií na platforme Java EE. Medzi najväčšie benefity používanie Spring Frameworku patrí [22]:

- Modularita
- Rieši len potrebné nové veci a na už vyriešené problémy využíva už existujúce technológie
- Jednoduchosť testovanie
- Vhodný návrh pre architektúru Model-View-Controller

4.8.1 Spring Framework Core

Jadro Spring Frameworku je postavené na princípe inverzie kontroly(*Inversion of Control (IoC)*) a aspektovo orientovanom programovaní (*Aspect-Oriented Programming (AOP)*). Inverzia kontroly, alebo tiež vkladanie závislostí (*Dependency Injection (DI)*), je proces, v ktorom objekt definuje svoje závislosti iba cez argumenty v konštruktore, cez argumenty vo factory metóde alebo cez vlastnosti, ktoré sú nastavené v inštancií objektu po jej vytvorení alebo po vrátení z factory metódy.[11]

4.8.2 Spring MVC

Spring Framework má svoju vlastnú nadstavbu pre návrhový vzor Model-View-Controller pre webové aplikácie a je súčasťou Spring Frameworku už od začiatku. Návrhovému vzoru Model-View-Controller sa budeme podrobnejšie venovať neskôr. Framework Spring MVC je postavený na Servlet API, konkrétne *DispatcherServlet*.[13] Ten sa stará o všetky HTTP požiadavky a odpovede a funguje na nasledovnom princípe [21]:

- Po prijatí HTTP požiadavky sa táto namapuje na vhodný Controller
- Controller zoberie požiadavku a zavolá vhodnú metódum ktorá vráti príslušný View
- DispatcherServlet predá všetky potrebné dáta z modelu a finálny View je vykreslený v prehliadači

4.8.3 Spring Security

Spring Security je silný a vysoko upraviteľný framework pre autentifikáciu a obmedzovanie prístupu. Je to v podstate štandard pre zabezpečovanie aplikácií založených na Spring-u. Táto nadstavba Spring-u sa zameriava na poskytnutie autorizácie aj autentifikácie v Java aplikáciách. Najväčšou výhodou je ľahká modifikácia a rozširovanie, aby boli splnené všetky požiadavky finálnej aplikácie. [23]

4.8.4 Spring Boot

Spring Boot je nadstavba Spring Frameworku, ktorá umožňuje jednoduché vytváranie samostatné spustiteľných aplikácií postavených na Spring-u. Väčšina aplikácií v Spring Boot-e vyžaduje veľmi malé množstvo dodatočných konfigurácií. Pomocou Spring Boot-u je možné vytvárať *jar* aplikácie, alebo tradičnejšie *war*, vhodné na nasadenie.[12]

4.9 Hibernate

Hibernate je framework na mapovanie databázových relácií na objekty (*Object-Relation Mapping (ORM)*). Zjednodušuje vývoj aplikácií, keďže nie je potrebné písať SQL dotazy do databázy, ktoré tento framework sám generuje. Ako nadstavbou svojho vlastného API poskytuje a implementuje aj špecifikáciu *Java Persistance API(JPA)*, takže je ľahko využiteľný a nasaditeľný na všetky druhy Java aplikácií, od JavaSE až po JavaEE aplikačné servery. Naviac je tento framework známy svojou spoľahlivosťou, kvalitou a výkonnosťou. [4]

4.10 MVC

Model-View-Controller (MVC) je jeden z najpoužívanejších návrhových vzorov. Zakladá sa na tom, že rozdeľuje aplikáciu na tri časti, model, view a controller. Model reprezentuje vlastnosti produktu a prepisuje business logiku, teda ako sa má s dátami pracovať. View vizualizuje reprezentáciu dát. Controller pracuje s modelom, vytvára jeho inštancie, vkladá do nich dáta a na základe požiadavok užívateľa zobrazuje príslušný View. Odpovedá teda na požiadavky používateľou naplnením potrebných modelov a zobrazením finálneho pohľadu užívateľovi. [3]

4.11 Možné alternatívy

Vhodnou alternatívou k použitému jazyku Java by mohol byť jazyk PHP, ktorý je taktiež vhodný na tvorbu informačných systémov. Je to serverový skriptovcí jazyk, ktorý sa najčastejšie využíva na tvorbu dynamických a interaktívnych webových stránok.[32]. Rovnako ako Java má priamu podporu pre viacero databázových systémov, ako napríklad MySQL, Oracle, Microsoft SQL Server a iné. [20] Rovnako ako JavaScript sa môže kód v jazyku PHP vkladať priamo do HTML stránok, ale narozdiel od JavaScriptu, ktorý sa interpretuje na strane klienta, PHP sa interpretuje na strane servera.

Keďže sa jedná o informačný systém na webovom rozhraní, frontend alebo vzhľad systému sa nedá definovať inak, ako pomocou jazykou HTML, CSS a Javascript.

Kapitola 5

Implementácia

V tejto kapitole sú postupne opísané procesy tvorby jednotlivých častí informačného systému, začínajúc od databázy cez implementáciu funkcionality systému až po jeho vzhľad.

5.1 Databáza

Keďže databáza je jednou z najdôležitejších častí informačného systému, jej návhu a implementácií sme sa venovali ako prvej. Na implementáciu databázy bol využitý nástroj *MySQL Workbench*, ktorý má priamu podporu pre jazyk SQL a taktiež MySQL. V tomto prostredí bol vytvorený skript `init.sql`, ktorý je súčasťou priloženého pamäťového média. Pomocou tohto skriptu sa najskôr vytvorí používateľ, samotná databáza a následne sa tomuto používateľovi pridelia práva pre prístup. Jednotlivé tabuľky databázy sa vytvoria automaticky z definovaných entít pri prvom spustení informačného systému.

Druhou časťou tvorby databázy bolo vytvorenie druhého skriptu `insert.sql`, ktorý slúži na vloženie počiatočných dát do databázy. Tieto dáta vychádzajú z reálnych údajov získaných počas analýzy výrobného procesu a slúžia na demonštráciu systému a následné testovanie. Tento skript sa taktiež nachádza na priloženom pamäťovom médiu.

5.2 Prepojenie databázy a informačného systému

Samotné parametre potrebné pre pripojenie k databáze sú definované v súbore `application.properties`. Je to špecifický súbor, ktorý sa využíva v Spring Boot-e na definovanie vlastností potrebných k behu aplikácie. Tento súbor sa musí nachádzať na ceste `src/main/resources`.

Databáza je na informačný systém napojená pomocou tried entít, ktoré sú namapované na jednotlivé tabuľky databázy. Toto mapovanie prebieha automaticky a je zaručené Spring anotáciami v triedach, ktoré určujú samotnú tabuľku (anotácia `@Table`) a tiež jednotlivé stĺpce tabuľky (anotácia `@Column`), na ktoré sa majú jednotlivé property mapovať. Každá takáto entita musí obsahovať anotáciu `@Entity` a taktiež musí obsahovať metódy `hashCode()` a `equals(Object obj)`, ktoré sa neskôr využívajú pri porovnávaní. Všetky entity sa nachádzajú v balíčku `sk.richardschleger.is.entity`.

5.3 Implementácia funkcionality systému

Funkcionalita systému je jeho najdôležitejšia časť. Preto je táto kapitola popísana podrobnejšie, navyše sú samostatne popísané prípady špeciálnej funkcionality, ktorá je v systéme implementovaná. Funkčnosť je rozdelená na viacero vrstiev, pričom každá z nich je popísaná v nasledujúcich sekciách.

5.3.1 Práca s databázou

Databáza je teda úspešne pripojená k informačnému systému a jej tabuľky sú namapované na jednotlivé entity. Aby sme však mohli s týmito entitami pracovať, je pre každú jednu z nich potrebný takzvaný *Data Access Object (DAO)*. Pomocou tohto objektu vieme získavať, pridávať, upravovať a odstraňovať dáta z databázy. Všetky tieto objekty sa nachádzajú v balíku `sk.richardschleger.is.dao`.

Implementácia DAO je veľmi jednoduchá. Musí obsahovať Spring anotáciu `@Repository`, ktorou dávame najavo, že táto trieda bude pracovať a pristupovať do databázy. Ďalej musí obsahovať property `EntityManager`, ktorý sa stará o životný cyklus samotných entít. Potom už v implemetácií ostávajú len jednotlivé metódy, napríklad pre získavanie všetkých dát `getAll()`, ktorý vracia list entít, `getById(int id)`, ktorý vráti entitu so zadaným identifikátorom, metóda `save(Entity entity)`, ktorá jednak pridáva nové entity do databázy, alebo ak už entita s daným identifikátorom existuje, aktualizuje ju, a poslednou metódou je metóda `remove(Entity entity)` alebo `remove(int id)`, ktoré odstránia zadanú entitu alebo entitu so zadaným identifikátorom zo systému. V implementácií všetkých metód je využívaná knižnica Hibernete, vďaka ktorej nie je potrebné písať SQL kód, ktorý táto knižnica automaticky generuje.

5.3.2 Pokročilá práca s dátami

Po vytvorení základných DAO objektov pre každú z entít bolo potrebné naimplementovať ďalšiu vrstvu, ktorá na svoje fungovanie využíva práve už vytvorené DAO objekty a navyše vykonáva požadovanú funkcionalitu. Práve toto zabezpečuje takzvaná *Service layer*, alebo vrstva služieb.

Každa implementácia služby pre niektorú z entít musí obsahovať Spring anotáciu `@Service`, ktorou oznamujeme Spring-u, že táto trieda implementuje služby. Jedna služba môže využívať viacero DAO objektov, teda môže pristupovať do dotabázy do rôznych tabuliek. Tieto DAO objekty sú automaticky mapované pomocou Spring anotácie `@Autowired`, teda ich nie je potreba inicializovať. Ďalej už nasledujú len implemetácie samotných metód. Pred každou metódou je ešte z bezpečnostných dóvodov Spring anotácia `@Transactional`, ktorá zabezpečuje transakčnosť dotazov do databázy. Je vhodnejšie túto anotáciu použiť vo vrstve služieb oproti DAO objektom z dôvodu, že jedna služba môže využívať viac ako jeden DAO objekt. Implementácie niektorých metód môžu byť veľmi jednoduché, môže sa v nich len zavolať príslušný DAO objekt a vykonať operáciu. Iné však vykonávajú veľké množstvo zložitejšej funkcionality, ako napríklad generovanie dokladov vo formáte PDF, ktoré je implementované v triede `DocumentServiceImpl`, alebo generovanie prehľadov používateľov alebo objednávok do formátu XLSX (Excel), ktoré je implemetované v triedach `UserServiceImpl` prípadne `OrderServiceImpl`. Tieto, ako aj ostatné triedy implementujúce vrstvu služieb sa nachádzajú v balíčku `sk.richardschleger.is.service`.

Generovanie dokladov vo formáte PDF

Na vytvorenie dokumentu vo formáte PDF a jeho následné naplnenie sme využili knižnicu iText. Je to open-source knižnica na vytváranie a konvertovanie PDF dokumentov.

Ako prvé je potrebné vytvoriť PDF dokument s požadovanými rozmermi. Tie zaistíme vytvorením objektu `new Rectangle(width, heigth)`, kde `width` je šírka dokumentu v bodoch a `height` je výška dokumentu taktiež v bodoch. Teraz môžeme vytvoriť nový iText Document a ako parameter mu predáme vytvorený objekt `Rectangle`. Ešte je potrebné vytvoriť objekt `ByteArrayOutputStream`, do ktorého bude výsledný dokument zapísaný. Teraz môžeme pristúpiť k samotnému naplneniu dokumentu.

Ešte pred samotným napĺňaním dokumentu je potrebné tento dokument otvoriť, a to metódou `document.open()`. Pre naplnenie dokumentu je potredný objekt typu `PDFWriter` ktorý získame pomocou metódy `PdfWriter.getInstance(doc, out)`, kde `doc` je vytvorený iText Document a `out` je výstupný ByteArrayOutputStream. Z tohto `PDFWriter`-u ešte potrebujeme získať `PDFContextByte` (ďalej len CB), ktorý získame použitím metódy `PDFWriter.getDirectContent()`. Teraz už pomocou získaného CB môžeme napĺňať dokument. Pomocou metódy `CB.setFontAndSize()` nastavíme požadovaný font a jeho veľkosť, ďalej metódou `CB.setTextMatrix(x,y)` nastavíme pozíciu, na ktorú sa má daný text zapísať a nakoniec metóda `CB.showText(text)` zobrazí zadaný text na nastavenú pozíciu daným fontom a veľkosťou.

Každý generovaný doklad ešte musí obsahovať čiarový kód, v ktorom je zakódované číslo vytvoreného dokladu. Tento kód je taktiež vytvorený pomocou knižnice iText, konkrétne sa jedná o objekt typu `Barcode128` (ďalej len barcode). Text, ktorý má byť v barcode zakódovaný, nastavíme pomocou metódy `barcode.setCode(text)`. Keďže v našom prípade tento kód obsahuje okrem číslic aj písmená, musím typ kódu nastaviť na `Barcode.CODE128` metódou `barcode.setCodeType(Barcode.CODE128)`. Teraz z generovaného barcode vytvoríme metódou `barcode.createImageWithBarcode(cb, null, null)` obrázok, ktorý môžeme vložiť do nášho iText Documentu. Nastavíme šírku a výšku obrázku pomocou metód `image.scaleAbsoluteWidth(width)` a `image.scaleAbsoluteHeight(height)`, metódou `image.setAbsolutePosition(x,y)` nastavíme jeho pozíciu a následné už len pomocou metódy iText Documentu `document.add(image)` vložíme obrázok do dokumentu.

Nakoniec je potrebné už len uzavrieť dokument metódou `document.close()` a následne vytvoriť z naplneného `ByteArrayOutputStream` objekt typu `ByteArrayInputStream`, ktorý je možné odoslať z controlleru do view. Tento proces je popísaný v kapitole 5.3.3.

Generovanie prehľadov vo formáte XLSX

Na tvorbu dokumentov programu Microsoft Excel vo formáte .XLSX sme využili knižnicu Apache POI, čo je open-source Java knižnica používaná na tvorbu, úpravu a zobrazovanie dokumentov Microsoft Office.

Na začiatku je potrebné vytvoriť objekt typu `Workbook`, konkrétne `XSSFWorkbook`. Po jeho vytvorení je potrebné pridať prázdny list, reporezentovaný objektom `Sheet`. Ten pridáme metódov `workbook.createSheet()`. Ďalej už len do vytvoreného listu musíme postupne vkladať jednotlivé riadky tabuľky. Tie vytvoríme metódov `sheet.createRow(r)`, kde `r` je číslo riadka, prvý riadok má číslo 0. Do riadku následne pridávame samotné bunky tabuľky. Bunku tabuľky vytvoríme pomocou metódy `row.createCell(c)`, kde `c` značí stĺpec, v ktorom má byť bunka umiestnená. Prvý stĺpec má taktiež číslo 0. Hodnotu bunky pridáme jej metódou `cell.setCellValue(value)`. Ďalej môžeme bunke priradiť štýl, ktorý priradíme metódou `cell.setCellStyle(style)`.

Štýl je reprezentovaný objektom `CellStyle` a vytvoríme ho metódou `workbook.createCellStyle()`. Štýlu môžeme priradiť napríklad zarovnanie textu metódou `style.setAlignment(alignment)` alebo font písma metódou `style.setFont`. Tento font vytvoríme pomocou metódy nami vytvoreného workbooku `workbook.createFont()`. Do tohto fontu môžeme následne priradiť vlastnosti ako veľkosť písma (`font.setFontHeightInPoints()`), farbu písma (`font.setColor(color)`) alebo hrúbku písma (`style.setBold(boolean)`).

Po úplnom naplnení môžeme pristúpiť k uloženiu dokumentu. v Našom prípade ho zapíšeme opäť do objektu typu `ByteArrayOutputStream`. Tento zápis vykonáme pomocou metódy `workbook.write(stream)`. Po úspešnom zapísaní ešte dokument uzavrieme metódou `workbook.close()`. Rovnako ako pri exporte dokumentov vo formáte PDF, aj tu vytvoríme objekt `ByteArrayInputStream` a ten vrátime do controlleru., ktorý ho odošle do view.

5.3.3 Predávanie dát do pohľadov (Modely a kontroléry)

O predávanie dát do pohľadov a ich následné zobrazenie sa starajú takzvané kontroléry. Ich úlohou je naplnenie príslušných modelov, predanie týchto modelov konkrétnemu pohľadu a následné zobrazenie tohto pohľadu.

Modely

Na to, aby bolo možné predať komplexnejšie dáta do pohľadu sú potrebné modely. Tieto sme implemetovali nie pre každú entitu, tak ako DAO alebo služby, ale len pre konkrétne prípady, kde sú tieto modely potrebné. Každý model pozostáva z vlastností (*properties*) a následných metód pre získavanie a nastavovanie hodnôt týchto vlastností. Keďže na rôzne situácie sú potrebné rôzne údaje, môže pre jednu entitu existovať aj viacero modelov, alebo naopak aj žiadny. Napríklad pre entitu `User` existujú modely pre pridávanie používateľa - `UserAddModel`, úpravu používateľa - `UserEditModel`, prehľad používateľov - `UserOverviewModel` a svojim spôsobom aj model pre výber používateľov pre prehľad - `SelectUsersModel`. Tieto, ako aj všetky ostatné modely sa nachádzajú v balíku `sk.richardschleger.is.model`.

Kontroléry

Implementácia každého z kontrolérov musí na začiatku obsahovať Spring anotáciu `@Controller`. Ďalej môže pred deklarovaním triedy obsahovať anotáciu `@RequestMapping(adresa)`, ktorá určuje požadované mapovanie url adresy. Túto anotáciu môže obsahovať rovnaku aj každá z metód v kontroléry. Jeden kontrolér môže pracovať s viacerými službami. Tieto sú automaticky mapované Springom pomocou anotácie `@Autowired` pred konštruktorom kontroléra. Ďalej už kontrolér obsahuje len implementované metódy, ktoré sú vo väčšine prípadov typu String, vrátia teda reťazec. Tento reťazec je následne doplnený o prefix a sufix, ktoré sú definované v súbore `application.properties`. Každá metóda, ktorá chce predať nejaké údaje pohľadu obsahuje parameter typu Model. Do tohto modelu sa pomocou jeho metódy `model.addAttribute(name, value)` vkladajú atribúty, ktoré sú následne prístupne z konkrétneho pohľadu. Týmito atribútmi môžu byť ďalšie modely alebo zoznamy, ktoré sú naplnené pomocou namapovaných služieb. Atribútom však môže byť aj samotná služba.

Špeciálnym prípadom metód kontrolérov sú metódy `generatePdf()` implementovaná v `SkladnikController` a metódy `generateOrderOverview` a `generateUserOverview` implementované v `VeduciController`. Tieto metódy nevracajú reťazec, ale sú typu `ResponseEntity`. Je to z dôvodu, že ich výsledkom nie je zobrazenie HTML stránky, ale v jednom prípade zobrazenie dokladu vo formáte PDF a v druhom prípade stiahnutie súboru vo formáte XLSX.

5.3.4 Zabezpečenie

Bezpečnosť a zabezpečenie systému zabezpečuje framework Spring spolu s jeho nadstavbou Spring Security. Nastavenia potrebné pre tento framework sú v triede `SecurityConfiguration`, ktorá je veľmi jednoduchá a obsahuje dve metódy `configure(AuthenticationManagerBuilder auth)` a `configure(HttpSecurity http)`. Prvá sa stará o to, aby sa údaje o používateľoch kontrolovali z pripojenej databázy. Druhá metóda nastavuje, do ktorých častí systému má prístup ktorá rola, taktiež určuje stránky na prihlásenie a špeciálnu stránku na zamietnutie prístupu.

Kým používateľ nie je prihlásený, má prístup len ku stránke na prihlásenie. Po zadaní údajov sú tieto porovnávané s tými uloženými v databáze. V databáze je uložené prihlasovanie meno v otvorenej podobe a heslo, ktoré je zašifrované pomocou algoritmu BCrypt. Ak sú zadané údaje správne, užívateľ je prihlásený do systému, v ktorom má následne prístup len do tých častí systému, ako mu to dovoľuje jeho rola. O odhlásenie používateľa sa taktiež automaticky stará Spring Security.

5.4 Implementácia vzhľadu systému (Pohľady)

Poslednou implementovanou časťou bola vrstva pohľadov, teda vzhľad celého systému. Keďže sa jedná o informačný systém na webovom rozhraní, na tvorbu pohľadov sme využili jazyk HTML spolu s jazykom CSS a JavaScript. Navyše k týmto technológiám je využívaná technológia *JavaServer Pages (JSP)*, ktorá pracuje na strane servera a dynamicky generuje k zadanej HTML stránke ďalšie údaje, elementy atď.

Pri tvorbe vizuálnej stránky systému bol kladený dôraz hlavne na prehľadnosť a jednoduchosť celého informačného systému. Všetky časti systému sú ladené do rovnakého štýlu, čo napomáha k celistvosti a jednotnosti. Zároveň je systém rozdelený do častí, z ktorých každá má vlastné jednoduché menu, aby sa predišlo jednému spoločnému navigačnému menu, ktoré by bolo zbytočne zložité. Na zobrazovanie viacerých údajov na stránke sú využívané prevažne tabuľky, pre vkladanie nových údajov alebo úpravu existujúcich sú využité vo veľkej miere formuláre. Po každej úspešne alebo neúspešne vykonanej operácií so systémom sa používateľovi zobrazí hláška, ktorá ho informuje o vykonanej operácií, respektíve chybe (napríklad *"Používateľ bol úspešne pridaný do systému!"*).

Keďže v informačnom systéme sa môže nachádať obrovské množtvo rôznych dát, na ich zobrazenie a prezentovanie používateľovi je vhodné využitie tabuliek, ktoré sú samy vhodne formátované. V našom informačnom systéme sme na zobrazovanie tabuliek využili knižnicu *DataTables*, čo je knižnica jazyka JavaScript a CSS, ktorá umožňuje tvorbu responzívnych, ľahko formátovateľných tabuliek. Navyše podporuje vyhľadávanie v tabuľkách, zobrazenie tabuľky je rozdelené na viacero stránok pre vyšší prehľad, je podporované radenie dát v tabuľke podľa ľubovoľného stĺpca a oveľa viac.

Pre pridávanie nových údajov do systému sú v pohľadoch vytvorené takzvané formuláre. Tieto formuláre sú pomocou `JSP` automaticky mapované na vložený model a po ich naplnení

sa v kontrolóroch znovu môže pracovať len z modelom, ktorý je naplnený automaticky vďaka tomuto mapovaniu. Rovnaký princíp platí aj pre úpravu dát, pri ktorej však v jednotlivých poliach budú už údaje predvyplnené podľa aktuálnych dát v databáze.

Kapitola 6

Testovanie

Testovanie je neoddeliteľnou súčasťou tvorby informačného systému. Cieľom testovania je včasné odhalenie prípadných chýb systému, ktoré by počas behu v reálnej firme mohli spôsobiť veľa problémov. Naše testovanie by sa dalo rozdeliť na dve hlavné časti, a to testovanie počas vývoja a finálne testovanie.

6.1 Testovanie systému počas vývoja

Súčasťou implementácie každej časti systému bolo jej podrobné testovanie na požadovanú funkčnosť. Toto testovanie by sme mohli rozdeliť na niekoľko častí, ako testovanie pripojenie databázy, následné testovanie jednotlivých služieb potrebných pre funkcionalitu, testovanie kontrolérov, ktoré sú potrebné pre vrátenie správneho pohľadu a nakoniec testovanie samotných pohľadov.

Pripojenie databázy bolo testované zároveň s implementovaním DAO objektov. V tej bolo už od začiatku uložených viacero dát, ktoré boli využité pre testovanie. Tieto údaje boli do databázy uložené pomocou skriptu `insert.sql`, ktorý sa nachádza na priloženom pamäťovom médiu. Keďže implemenácia DAO objektov je veľmi jednoduchá, aj testovanie prebiehalo veľmi jednoducho a bez problémov. Testovanie prebiehlo zväčša manuálne kontrolovaním vrátených údajov a porovnaním týchto údajov s dátami v databáze.

Ďalšou časťou bolo testovanie služieb. To prebiehalo podobne ako testovanie databázy, teda manuálne kontrolou dát. Keďže služby sú oveľa komplexnejšie ako DAO objekty, aj k ich testovaniu sme pristupovali podrobnejšie. Bolo potrebné otestovať oveľa väčšie množstvo možných scenárov, ktoré pri používaní systému môžu nastať a ako vrstva služieb bude na tieto scenáre reagovať. Keďže používateľ nie vždy zadáva len platné údaje a tak nie vždy sa do vrstvy služieb dostanú korektné údaje, testovali sme aj tieto možnosti zadávania nezmyselných dát, s ktorými sa vrstva služieb musí vedieť vysporiadať. Počas implementácie sa takto podarilo odhaliť a opraviť veľké množstvo chýb, ktoré by sa inak premietli do kontrolérov a tak aj do výsledného pohľadu používateľovi.

Posledné dve časti testovania, teda testovanie kontrolérov a výsledných pohľadov, sú veľmi úzko prepojené, keďže kontroléry sa starajú o naplnenie a zobrazenie korektných pohľadov. Pri kontroléroch sme teda testovali hlavne ich schopnosť správne naplniť modely získanými dátami z databázy a ich následnú schopnosť správne zareagovať na získané modely naplnené prácou používateľa so systémom. Pri testovaní sa vyskytlo len zopár častí, s ktorými bolo trochu viac práce ako s ostatnými. Konkrétne šlo o vrátenie správneho dokladu vo formáte PDF alebo vrátenie prehľadu vo formáte XLSX. Ostatné časti fungovali

zväčša správne alebo bolo ich chovanie veľmi rýchlo a jednoducho opravené. Pri testovaní pohľadov sme skúmali hlavne korektnosť zobrazovania dát z modelov, zobrazovanie informačných správ používateľovi o vykonaných operáciách v systéme a možnosť zadávania nových dát do systému pomocou formulárov. Všetky časti boli opravené a doladené do požadovanej funkčnosti. Zobrazovanie dát bolo kontrolované na platforme počítačov, keďže sa nepredpokladná práca so systémom na iných zariadeniach. Testovanie prebiehalo vo všetkých najpoužívanejších prehliadačoch ako:

- Microsoft Edge
- Mozilla Firefox
- Google Chrome
- Opera

6.2 Finálne testovanie

Finálne testovanie malo prebiehať v spoločnosti Tesla Liptovský Hrádok súbežne s behom aktuálne používaného systému. Avšak aktuálna situácia s vírusom COVID-19 toto testovanie neumožnila, preto finálne testovanie nakoniec prebiehalo v simulovanom prostredí v spolupráci s riaditeľom divízie elektro Tesly Liptovský Hrádok na reálnych dátach, ktoré mi boli touto firmou poskytnuté.

V tejto časti testovania sa už predpokladala správna funkčnosť systému a teda testovanie prebiehalo formou reálnej práce so systémom. Išlo o úkony ako:

- Pridať, upraviť a odstrániť zamestnanca v systéme
- Pridať nový sklad, následne ho upraviť a nakoniec odstrániť
- Pridať do systému novú objednávku na určitý výrobok
- Odhlásiť výrobu pre túto objednávku
- Expedovať zhotovené výrobky a vytlačiť sprievodný doklad
- Vytvoriť prehľad zamestnancov
- atď.

Pri všetkých operáciách, na ktoré je systém v aktuálnej verzií implementovaný, sa systém choval podľa očakávaní, teda používateľovi zobrazoval správne údaje, ukladal do databázy nové, upravoval a odstraňoval existujúce. Informačný systém teda prešiel finálnym testovaním bez väčších problémov.

Kapitola 7

Záver

Cieľom tejto bakalárskej práce bolo navrhnúť a následne implementovať informačný systém pre riadenie výroby na webovom rozhraní. Pre kvalitný návrh systému bola potrebná hĺbková analýza procesu výroby a riadenia skladov v špecialitovanej firme a taktiež preštudovanie aktuálnej softvérovej podpory v tejto oblasti. Následný návrh systému prebiehal na základe získaných údajov z výrobnej firmy. Implementácia postupovala bez zásadných problémov a pravidelne sa testovali jednotlivé časti systému na požadovanú funkčnosť.

Po úspešnom prejdení finálneho testovania v simulovaných podmienkach sa dá povedať, že systém spĺňa všetky požiadavky naň kladené. Určite by bolo vhodné dlhodobejšie testovanie v reálnych podmienkach výrobnej firmy, ktoré sa však nemohlo uskutočniť kvôli situácií s vírusom COVID-19. Zo simulovaných testov sa systém zdá pripravený na reálne použitie vo firmách zameraných na výrobu vlastných výrobkov, ktoré následne exportujú svojim zákazníkom.

Ďalšie možné rozšírenie systému vidím v ekonomickej stránke riadenia firmy, kde v systéme by bolo možné priamo nakupovať materiál, komunikovať so zákazníkmi, vystavovať faktúry a tak ďalej.

Literatúra

[1] BEGINNER-SQL-TUTORIAL.COM. *SQL Commands* [online]. [cit. 2020-03-06]. Dostupné z: https://beginner-sql-tutorial.com/sql-commands.htm.

[2] BOOTSTRAP GITHUB TEAM. *Bootstrap releases* [online], 28. novembra 2019 [cit. 2020-03-05]. Dostupné z: https://github.com/twbs/bootstrap/releases.

[3] GROCHOĽ, L. *Po lopate: Čo je to MVC (Model – View – Controller)* [online], 24. mája 2019 [cit. 2020-03-10]. Dostupné z: https://blog.bart.sk/co-je-mvc-model-view-controller-po-lopate/.

[4] HIBERNATE. *Hibernate ORM* [online]. [cit. 2020-03-10]. Dostupné z: https://hibernate.org/orm/.

[5] JANOVSKÝ, D. *CSS styly - úvod* [online]. [cit. 2020-03-04]. Dostupné z: https://www.jakpsatweb.cz/css/css-uvod.html.

[6] JOHARI, A. *What Is Java? A Beginner's Guide to Java and Its Evolution* [online]. 2020-02-27 [cit. 2020-03-06]. Dostupné z: https://www.edureka.co/blog/what-is-java/.

[7] MDN CONTRIBUTORS. *HTML: Hypertext Markup Language* [online]. 2020-01-18 [cit. 2020-03-04]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML.

[8] MDN CONTRIBUTORS. *JavaScript* [online]. 2020-03-20 [cit. 2020-03-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[9] ORACLE CORPORATION. *Chapter 11 Data Types* [online]. [cit. 2020-03-06]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/data-types.html.

[10] ŠPIRENG, P. *Java EE – sprievodca úplných začiatočníkov* [online], 4. januára 2014 [cit. 2020-03-07]. Dostupné z: https://robime.it/java-ee-sprievodca-uplnych-zaciatocnikov/.

[11] PIVOTAL, INC.. *Core Technologies* [online]. 5.2.5.RELEASE. [cit. 2020-03-08]. Dostupné z: https://docs.spring.io/spring/docs/current/spring-framework-reference/core.html.

[12] PIVOTAL, INC.. Introducing Spring Boot. *Spring Boot Reference Documentation* [online]. 2.2.5.RELEASE. [cit. 2020-03-08]. Dostupné z: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#getting-started-introducing-spring-boot.

[13] Pivotal, Inc.. Spring Web MVC. *Web on Servlet Stack* [online]. 5.2.5.RELEASE. [cit. 2020-03-08]. Dostupné z: https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html#mvc.

[14] Čápka, D. Charakteristiky podnikových aplikací. *Úvod do Java Enterprise Edition (JEE)* [online]. [cit. 2020-03-07]. Dostupné z: https://www.itnetwork.cz/java/jee/java-enterprise-edition-uvod-do-jee-j2ee.

[15] QuinStreet Inc.. *What is SQL?* [online]. [cit. 2020-03-06]. Dostupné z: http://www.sqlcourse.com/intro.html.

[16] Raffai, Z. *Different Java platforms* [online], 3. marca 2018 [cit. 2020-03-06]. Dostupné z: https://www.zoltanraffai.com/blog/different-java-platforms/.

[17] Sun Microsystems, Inc.. Design Goals of the Java TM Programming Language. *The Java Language Environment* [online]. 1997 [cit. 2020-03-06]. Dostupné z: https://www.oracle.com/technetwork/java/intro-141325.html.

[18] szj.cz. *Vlastnosti typových systémů* [online]. [cit. 2020-03-05]. Dostupné z: http://szj.cz/tag/silne-a-slabe-typovane-jazyky/.

[19] TIOBE. *TIOBE Index for March 2020* [online]. [cit. 2020-03-05]. Dostupné z: https://www.tiobe.com/tiobe-index/.

[20] Tutorials Point. Why to learn PHP? *PHP Tutorial* [online]. [cit. 2020-04-12]. Dostupné z: https://www.tutorialspoint.com/php/index.htm.

[21] Tutorials Point. *Spring - MVC Framework* [online]. [cit. 2020-03-08]. Dostupné z: https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm.

[22] Tutorials Point. *Spring Framework - Overview* [online]. [cit. 2020-03-08]. Dostupné z: https://www.tutorialspoint.com/spring/spring_overview.htm.

[23] VMware, Inc.. *Spring Security* [online]. [cit. 2020-03-08]. Dostupné z: https://spring.io/projects/spring-security.

[24] W3Schools. What is Bootstrap? *Bootstrap Get Started* [online]. [cit. 2020-03-05]. Dostupné z: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp.

[25] W3Schools. CSS Syntax. *CSS Syntax* [online]. [cit. 2020-03-04]. Dostupné z: https://www.w3schools.com/css/css_syntax.asp.

[26] W3Schools. A Simple HTML Document. *HTML Introduction* [online]. [cit. 2020-03-05]. Dostupné z: https://www.w3schools.com/html/html_intro.asp.

[27] W3Schools. SQL is a Standard - BUT.... *Introduction to SQL* [online]. [cit. 2020-03-06]. Dostupné z: https://www.w3schools.com/sql/sql_intro.asp.

[28] W3Schools. *JavaScript Data Types* [online]. [cit. 2020-03-05]. Dostupné z: https://www.w3schools.com/js/js_datatypes.asp.

[29] W3Schools. What is jQuery? *JQuery Introduction* [online]. [cit. 2020-03-05]. Dostupné z: https://www.w3schools.com/jquery/jquery_intro.asp.

[30] W3Schools. jQuery Syntax. *JQuery Syntax* [online]. [cit. 2020-03-05]. Dostupné z: https://www.w3schools.com/jquery/jquery_syntax.asp.

[31] W3Schools. What is MySQL? *PHP MySQL Database* [online]. [cit. 2020-03-06]. Dostupné z: https://www.w3schools.com/php/php_mysql_intro.asp.

[32] W3Schools. *PHP Tutorial* [online]. [cit. 2020-04-12]. Dostupné z: https://www.w3schools.com/php/.

Príloha A

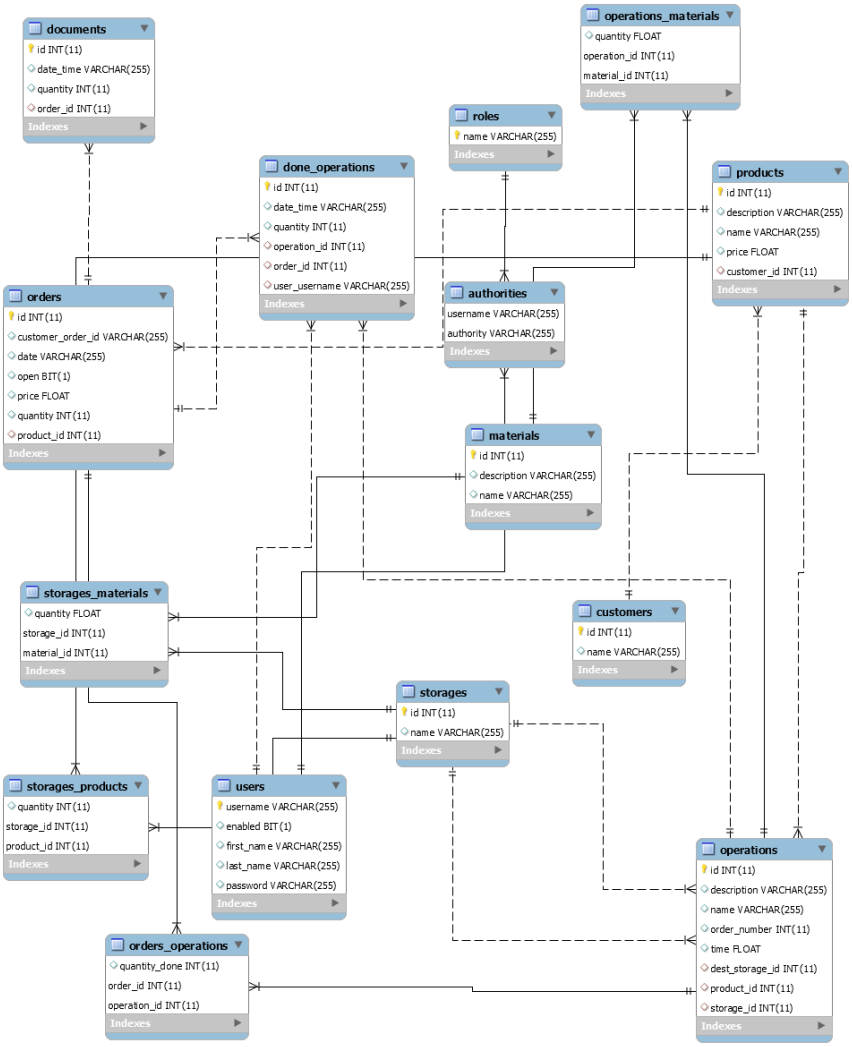
Obsah priloženého pamäťového média

Na priloženom CD sa nachádzajú nasledujúce položky:

- is/src/ - zdrojové súbory potrebné pre preloženie aplikácie
- is/target/ - preložená aplikácia, nachádza sa tu spustiteľný súbor is-1.0.war
- is/pom.xml - konfiguračný súbor pre preklad pomocou nástroja MAVEN
- latex/ - zdrojové súbory využíté pre vytvorenie technickej správy bp.pdf
- bp.pdf - bakalárska práca vo formáte pdf
- README.md - návod na úspešný preklad a spustenie aplikácie

Príloha B

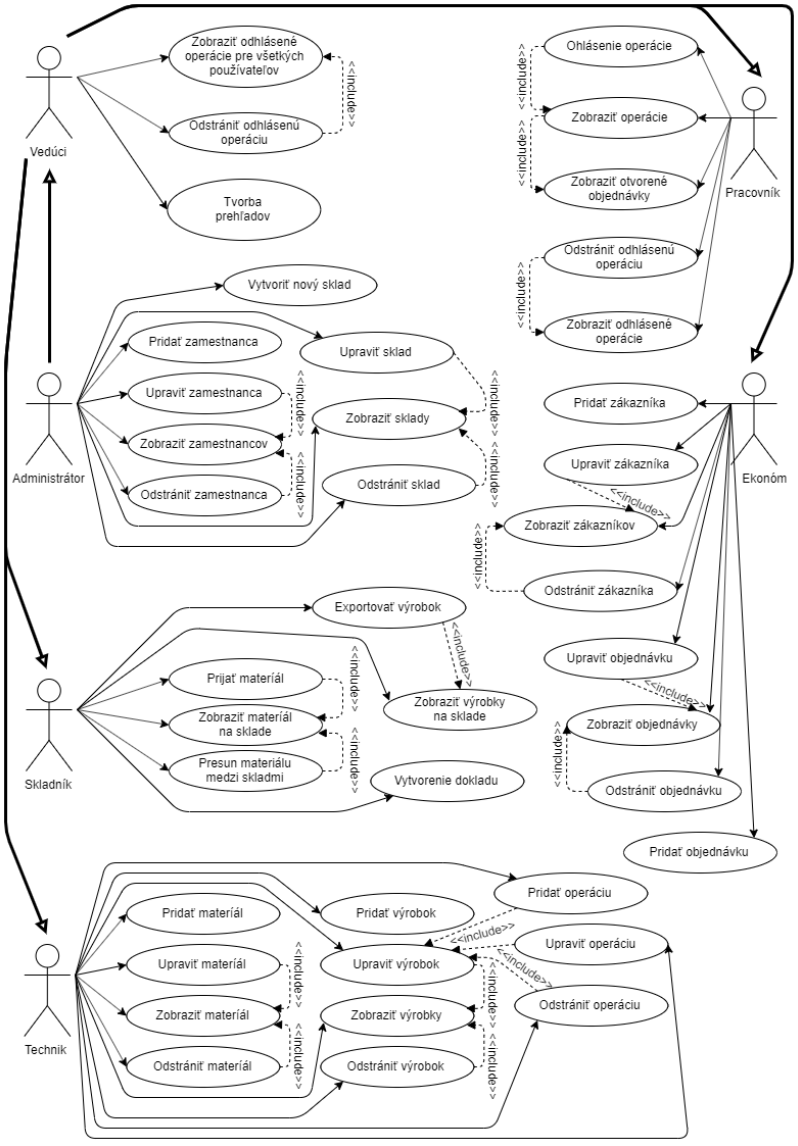
Entity Relation Diagram



Obr. B.1: ER diagram databázy informačného systému

Príloha C

Diagram prípadov použitia



Obr. C.1: Diagram prípadov použitia informačného systému

Príloha D

Návod k inštalácii a spusteniu systému

Prvé spustenie

1. Pred prvým spustením je potrebná inicializácia databázy. To dosiahneme spustením skriptu `init.sql` nachádzajúcom sa v priečinku `src/sql`.

2. Pre preklad zdrojového kódu je potrebný nástroj MAVEN a Java Development Kit vo verzií 8 a vyššie. Pre následný beh programu je potrebný bežiaci MySQL server. Parametre pripojenia k tomuto serveru sa nachádzajú v súbore `application.properties`, ktorý sa nachádza v priečinku `src/main/resources`.

3. Preklad spustíme príkazom „mvn clean package“ spusteným v priečinku, kde sa nachádza súbor `pom.xml`. Tým sa preloží zdrojový kód do priečinku target a vytvorí sa spustiteľný súbor `is-1.0.war`, ktorý obsahuje všetky potrebné súbory pre beh informačného systému.

4. Spustíme vytvorený súbor príkazom „java -jar is-1.0.war“ spusteným v priečinku `target`.

5. Pre prípady testovania sa v priečinku `src/sql` nachádza aj druhý skript, `insert.sql`, ktorý napĺňa túto databázu testovacími dátami. Pred spustením tohto skriptu musí byť aspoň raz spustený samotný informačný systém, čo zaručí vygenerovanie tabuliek databázy.

Opätovné spustenie

1. Ak už máme zdrojový kód preložený, teda v priečniku `target` sa nachádza spustiteľný súbor `is-1.0.war`, stačí tento súbor spustiť príkazom „java -jar is-1.0.war“ spusteným v priečinku `target`.