МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ОЛІЙНИК Ярослав Петрович

Програмна система управління запасами на підприємстві/ Software system for inventory management in the enterprise

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконав студент групи ПЗС-42
Я. П. Олійник

Науковий керівник:
викладач ВЕРЕМЧУК А.В.

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

ЗМІСТ

ВСТУП

Розвиток ринкових відносин в XXI і посилення конкуренції серед підприємств ставить економічні суб'єкти в жорсткі умови господарювання, що вимагає від них високоякісного управління всіма процесами і грамотного розпорядження фінансовими і матеріальними ресурсами. У міру зміни економічних умов всі підприємства стикаються з необхідністю вдосконалення своїх економічних структур.

Витрати, пов'язані з запасами, є однією з основних складових собівартості продукції (наприклад, частка витрат на запаси в собівартості продукції машинобудування доходить до 60%), тому процеси управління запасами є важливою складовою частиною системи управління підприємством. Ефективність цих процесів характеризується таким ключовим критерієм, як величина витрат, які утворюються при управлінні запасами. З точки зору практики проблема управління запасами є надзвичайно серйозною. Втрати, які несуть підприємства внаслідок нераціонального управління запасами, дуже великі. Погано, коли запас малий, недостатній. Це може привести до порушення ритмічності виробництва, зростання собівартості продукції, зриву термінів виконання робіт за договорами, втрати прибутку і репутації компанії - штрафу за незадоволений або відкладений попит. Однак, вкрай небажаною є й ситуація, коли запас надмірно великий. В цьому випадку відбувається "заморожування" оборотних коштів організації, ростуть витрати на зберігання запасів.

Також важливим питанням є необхідна величина запасних частин, які підприємство створює на випадок непередбачених збоїв з постачанням при виробництві, поломки інструментів, обладнання. Очевидно, що запасні частини є додатковими запасами (запасні матеріали, деталі, інструменти, комплектуючі вироби та запасні частини для обладнання), які погіршують фінансові результати виробничої діяльності (за рахунок заморожування коштів у запасах), але забезпечують підприємству стійкість і ліквідність, тому проблема

застосування моделей і систем управління запасами для розрахунку раціональної кількості запасів є актуальною в даний час.

В даний час рішення задач підвищення ефективності управління підприємством в цілому неможливо без застосування сучасних обчислювальних систем і програмних комплексів. На українському ринку в останні роки представлено досить велика кількість програмних продуктів, що сприяють підвищенню якості та ефективності процесів постачання споживачів різною продукцією.

Однак в переважній більшості наявних на ринку програмних продуктів добре представлені функції обліку і руху товарів і, практично, відсутні функції за кількісним розрахунком раціональних обсягів запасів. Це значно знижує ефективність від результатів роботи існуючих програмних продуктів.

Звідси випливає необхідність дослідження і створення на підприємствах таких систем управління запасами, які б враховували особливості вітчизняного виробництва, головною з яких, в даному випадку, є позамовний характер виготовлення продукції.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПІДТРИМКИ ПРОЦЕСІВ УПРАВЛІННЯ ЗАПАСАМИ НА ПІДПРИЄМСТВІ

1.1. Коротка характеристика об’єкту управління

Корпорація «Богдан», одна з найбільш динамічних за розвитком компаній в Україні, об’єднує в собі потужності для виробництва автобусів і тролейбусів, легкових авто, вантажної та комерційної техніки, а також має власну розгалужену торгівельно-сервісну мережу.

Виробничі потужності корпорації «Богдан», в створення яких інвестовано понад 440 млн. доларів США, дозволяють сьогодні виготовляти 120-150 тис. легковиків, до 9 тис. автобусів та тролейбусів у всіх класах, а також біля 15 тис. вантажівок та спеціалізованої техніки. Заводи компанії розташовано у Луцьку, Черкасах та Криму. Усі виробничі процеси максимально автоматизовані. Новітнє обладнання виробничих ліній від світових виробників повністю універсальне, що дозволяє виготовляти будь які авто за марками та габаритами.

Торгівельно-сервісна мережа корпорації представлена одним з найбільших операторів українського ринку – "Богдан-Авто Холдинг", який має філії в усіх регіонах країни. Окрім розвитку торгівельної мережі, корпорація особливу увагу приділяє сервісу. Всі філії "Богдан-Авто Холдинг" оснащені сучасним технологічним обладнанням для сервісного, гарантійного і післягарантійного обслуговування автомобілів.

Державна реєстрація юридичної особи - 04.11.1996 року.

1955-Луцький ремонтний завод ввели в експлуатацію;

1959 - завод з ремонтного перетворюється на машинобудівний;

1967 - розпочато серійний випуск автомобілів;

1995 - Луцький автомобільний завод із державного перетворюється у відкрите акціонерне товариство ВАТ "ЛуАЗ"

2000 - розпочалося великовузлове складання автомобілів ВАЗ-21093 і УАЗ різних модифікацій;

- укладена угода по співпрацю з корпорацією «Богдан»

- початок програми з випуску автобусів та тролейбусів.

Юридична: 43010, Волинська область м.Луцьк, вул.Рівненська, 42 Фактична : 43010, Волинська область м.Луцьк, вул.Рівненська,42.

ПІБ керівника ПОУ, телефон: Джула Микола Михайлович, 0332788244

ПІБ керівника відділу кадрів, тел.: Гань Лариса Володимирівна, 0332784122

Транспортна доступність (фрагмент мапи міста, селища, району із відображенням маршрутів громадського транспорту, в тому числі доставки працівників з інших населених пунктів): Тролейбус № 1, маршрутні таксі № 15,15а, 17а, 2,16,29,12,10 . Підприємство надає власне транспортне забезпечення для жителів інших населених пунктів Волині.

Вид економічної діяльності (відповідно до Державного класифікатора видів економічної діяльності) Виробництво дорожніх транспортних засобів 2.2. Види товарів, що виробляються або послуг, що надаються : Виробництво дорожніх транспортних засобів (тролейбусів та автобусів «Богдан»)

Рис. 1.1. Виробництво дорожніх транспортних засобів

Основні види обладнання та устаткування : лазерна порізка металу «Амада», верстати з числовим програмним керуванням, конвеєрні лінії для складання транспортних засобів, фарбувальні камери «Фірат», фарбувальний комплекс «Ейзенман».

Професійна структура. Чисельність працюючих - 625 осіб. Професійний склад працівників: електрозварник на автоматичних та напівавтоматичних машинах; електромонтер з ремонту та обслуговування електроустаткування; зварник на машинах контактного (пресового)зварювання; різальник металу на ножицях і пресах; різальник на пилах, ножівках та верстатах; слюсар з механоскладальних робіт; слюсар з ремонту автомобілів; стропальник; токар; фрезерувальник; маляр;

Режим праці. Змінність (по окремим підрозділам, виробництвам, групам професій): І зміна, ІІ зміни (планується ввести ІІІ зміни) Обідні та технологічні перерви - при 8-год. р.д. : обідня перерва - 0,5 год.; тех.перерви - 2 по 10 хв. При 12-год.р.д.: 2 обідні перерви по 0,5 год.; тех.перерви - 3 по 10 хв. Тривалість та особливості надання відпусток - 24к.д., 24 к.д.+ за шкідливі умови праці, або використання понаднормового часу(4-7 к.д.). Застосування понаднормового часу - передбачено з додатковою оплатою.

Заробітна плата. Система оплати праці (по окремих підрозділах, виробництвах, групах професій) - робітники - відрядно-преміальна. Середня заробітна плата (по окремим підрозділам, виробництвам, групам професій) - робітники – 4500 грн.; службовці – 6000 грн.

Рис. 1.2. Виробництво тролейбусів

Можливості професійного навчання:

- підготовка (в тому числі на робочому місці);
- підвищення кваліфікації

Навчальний центр підготовки, перепідготовки та підвищення кваліфікації ДП «АСЗ № 1» - підготовка кадрів на виробництві для потреб підприємства.

Соціальна інфраструктура підприємства (наявність):

- Гуртожиток - місто Луцьк вул.. Дубнівська, 31 в. Житлове будівництво - не передбачено. Лікувально-оздоровчі заклади : наявний медичний пункт. Дитячі виховні заклади : відсутні. Культурно-освітні заклади - є бібліотека. Побутове обслуговування : на території діє їдальня. Пільги, в тому числі згідно з колективним договором (часткове відшкодування проїзду, гнучкий графік роботи, преміювання згідно колективного договору).

1.2. Опис предметної області

Формування товарних запасів дозволяє підприємству забезпечувати стійкість асортименту товарів, здійснювати певну цінову політику, підвищувати рівень обслуговування покупців. Все це вимагає підтримки на кожному підприємстві оптимального рівня запасів по кожній товарній позиції.

Товарні запаси підприємств торгівлі перебувають у постійному русі та оновлення. Кінцевою стадією їх руху є споживання. Основним призначенням товарних запасів в оптовій торгівлі є обслуговування оптових покупців (роздрібне торговельне підприємство), а в роздрібному - забезпечення стійкості пропозиції товарів споживачам. Для підтримки товарних запасів на оптимальному рівні необхідна чітко налагоджена система управління запасами. Оптимальний рівень запасів означає такий стан, коли відсутні надлишки запасів або їх дефіцит.

Управління товарними запасами передбачає наступні операції:

1. Нормування запасів означає вироблення економічно обґрунтованих нормативів. Приміром, обсяг страхового запасу. Нормативи розраховуються окремо для поточного запасу, страхового запасу, сезонного зберігання і т. д.

2. Оперативний облік і контроль. Здійснюється за допомогою спеціального програмного забезпечення. До прикладу. 1С Торгівля і склад. Залишки товарів на початок і кінець місяця піддаються аналізу і коригування.

3. Регулювання. Полягає у підтримці їх на певному рівні і зміну їх величини в залежності від попиту і періодів поставки товарів. Як надлишок, так і недолік запасів надають негативну дію на результати комерційної діяльності підприємства

При постановці системи ефективного управління запасами на підприємстві необхідно в першу чергу врівноважити дві чаші віртуальних ваг. На одній з них слід зосередити позитивні сторони наявності запасів - забезпечення більшої надійності в роботі, вираженої в безперервності виробничого процесу і (або) задоволення потреб покупців, а на інший - негативні: витрати на утримання (зберігання) запасів і відволікання з обороту капіталу, інвестованого в запаси.

Розробка системи управління запасами на підприємстві ведеться з урахуванням стратегічних пріоритетів компанії. При цьому встановлюється компроміс між ризиками і витратами або ліквідністю і оборотністю.

Рекомендується п'ять послідовних етапів постановки системи управління запасами на підприємстві:

Етап 1. Визначення вартості запасів, їх номенклатури і кількісних характеристик, тобто об'ємних, часових параметрів, дані про місцезнаходження.

Етап 2. Проведення АВС-аналіз а і виявлення ключових запасів категорії "А", менш важливих - категорії "В" - і другорядних за значенням - категорії "С".

Етап 3. Реєстрація методів і процедур, що використовуються компанією в даний час при управлінні запасами. Вибір критеріїв для оцінки результативності існуючої системи управління запасами та постановка обліку, що дозволяє одержувати всю необхідну інформацію для цієї мети.

Етап 4. Порівняння існуючих методів і процедур управління з необхідними. Налагодження системи інформаційного моніторингу запасів, ходу виконання замовлень, витрат по зберіганню запасів.

Етап 5. Визначення кроків переходу до нової системи управління запасами. Розробка нової або удосконалення діючої системи управління запасами.

Оптимальне управління запасами передбачає отримання чіткої відповіді на два основних питання: коли треба розпорядитися про поповнення запасу, скільки при цьому треба замовляти матеріальних запасів, що йдуть у запас.

Існують дві основні моделі управління запасами: система з фіксованим обсягом ярма розміром замовлення, звана також моделлю економічного розміру замовлення, або Q-моделлю, і система з фіксованою періодичністю замовлення, звана періодичної моделлю, або Р-моделлю.

Сутність Q-моделі полягає в тому, що як тільки запас якого-небудь товару досягне заздалегідь певного мінімального значення ярма точки замовлення, цей товар замовляється. Досягнення мінімального рівня може виникнути в будь-який момент і залежить від інтенсивності попиту.

Для пояснення сутності цієї моделі можна провести аналогію з пляшкою рослинної олії, що використовується в домашньому господарстві. Кожен раз в магазині купується однаковий об'єм масла - 1 літр (фіксований обсяг замовлення). Як тільки рівень масла в пляшці досягне певного рівня (рівень, відповідний близько 100 мл), купується чергова упаковка.

У практиці управління запасами Q-модель використовується в наступних випадках:

♦ великі втрати в результаті відсутності запасу;

♦ високі витрати по зберіганню запасу;

♦ висока вартість товару, що замовляється;

♦ висока ступінь невизначеності попиту.

Використання Q-моделі передбачає постійний контроль залишку запасів. Ця модель вимагає, щоб кожен раз, коли проводиться вилучення ресурсів із запасу, виконувалася перевірка, чи досягнута крапка чергового замовлення.

Оптимальний розмір партії товарів, що поставляються, і оптимальна частота завозу залежать від наступних факторів:

♦ обсягу попиту;

♦ витрат по доставці товарів;

♦ витрат по зберіганню запасу.

В якості критерію оптимальності вибирають мінімум сукупних витрат по доставці і зберіганню.

При управлінні запасами по Р-моделі період, через який підприємство направляє замовлення постачальнику, залишається незмінним. Наприклад, кожен понеділок менеджер фірми переглядає залишки товарів і доводить їх до заздалегідь визначеної максимальної норми. Розмір замовленої партії товару визначається різницею передбаченого нормою максимального товарного запасу та фактичного запасу. Оскільки для виконання замовлення потрібно оптимальний період часу, то величина замовленої партії збільшується на розмір очікуваної витрати на цей період.

Система контролю за станом запасів з фіксованою періодичністю замовлення застосовується в наступних випадках:

♦ умови постачання дозволяють отримувати замовлення різними по величині партіями;

♦ витрати з розміщення замовлення і доставку порівняно невеликі;

♦ втрати від можливого дефіциту незначні

При використанні тих чи інших методів управління запасами необхідно обов'язково враховувати особливості попиту на продукцію підприємства, а також особливості локальних або загальносистемних інформаційних технологій, які забезпечують автоматизоване управління бізнесом

Прийняття оптимального рішення по управлінню запасами потребує врахування багатьох чинників і завжди повинно спиратися на пошук

логістичного компромісу, забезпечує поряд з скороченням витрат повне задоволення попиту на необхідну продукцію.

На практиці складність у виборі певного підходу залежить від умов, в яких функціонує підприємство, і здібності менеджера з логістики прийняти потрібне рішення. Чим складніше умови, тим більш складна модель управління запасами потрібно. Для всіх моделей характерна наявність двох серйозних проблем: забезпечення належного контролю за кожним елементом запасу та гарантування точного відстеження стану готівкових запасів.

До найбільш поширеним додатковим системам управління запасами відносяться:

♦ система з встановленою періодичністю поповнення запасів до певного рівня:

♦ система "мінімум-максимум".

В системі з встановленою періодичністю поповнення запасів до певного рівня, як і в системі з фіксованим інтервалом часу між замовленнями, вхідним параметром є період часу між замовленнями. На відміну від основної системи, вона зорієнтована на роботу при значних коливаннях споживання. Щоб запобігти завищенню обсягів запасів, які знаходяться на складі, або їх дефіциту, замовлення виробляються не тільки у встановлені моменти часу, але і за досягнення запасом граничного рівня. Система включає в себе елемент P-моделі, тобто встановлену періодичність оформлення замовлення, і елемент Q-моделі, тобто відстеження граничного рівня замовлення.

Відмінною особливістю системи є те, що замовлення поділяються на дві категорії - планові і додаткові. Планові замовлення виробляються через задані інтервали часу, а додаткові замовлення - відхилення темпів споживання від запланованих.

Система "мінімум-максимум" також містить у собі елементи основних систем управління. Система орієнтована на ситуацію, коли витрати на облік запасів і витрати на оформлення замовлення настільки значні, що стають порівнянні з втратами від дефіциту запасів. Тому в даній системі замовлення

виробляються не через кожен заданий інтервал часу, а тільки за умови, що запаси на складі в цей момент виявилися рівними або меншими встановленого мінімального рівня. У цьому випадку розмір розраховується так. щоб постачання поповнило запаси до максимально бажаного рівня. Таким чином, система працює лише з двома рівнями запасів - мінімальним і максимальним.

1.3. Огляд і аналіз існуючих аналогів, що реалізують функції предметної області

Для огляду функціональності ERP-систем, були взяті такі, найбільш поширені ERP-системи на українському ринку IT-продуктів:

- 1С;
- SAP;
- Галактика;
- Oracle;
- MS Axapta;
- RS-Balance 3;

У кожній із зазначених систем виділялися модулі, в яких ведеться управління запасами матеріалами, комплектуючими, виробами. Джерелами інформації для аналізу функцій перерахованих систем виступали різні презентації продуктів, технічна документація, посібники.

Огляд функцій модуля управління запасами в системі 1С (рисунок 1.3):

- Облік надходжень запасів (матеріалів, комплектуючих) по складах;
- Облік накладних витрат;
- Оперативний складський облік запасів різного роду (прихід, розхід, сортування);
- ефективне управління пересуванням і розподілом матеріальних ресурсів;
- інформаційне забезпечення складських і логістичних процесів;
- ведення ордерного складського обліку;

• адресне зберігання товарів на складі;

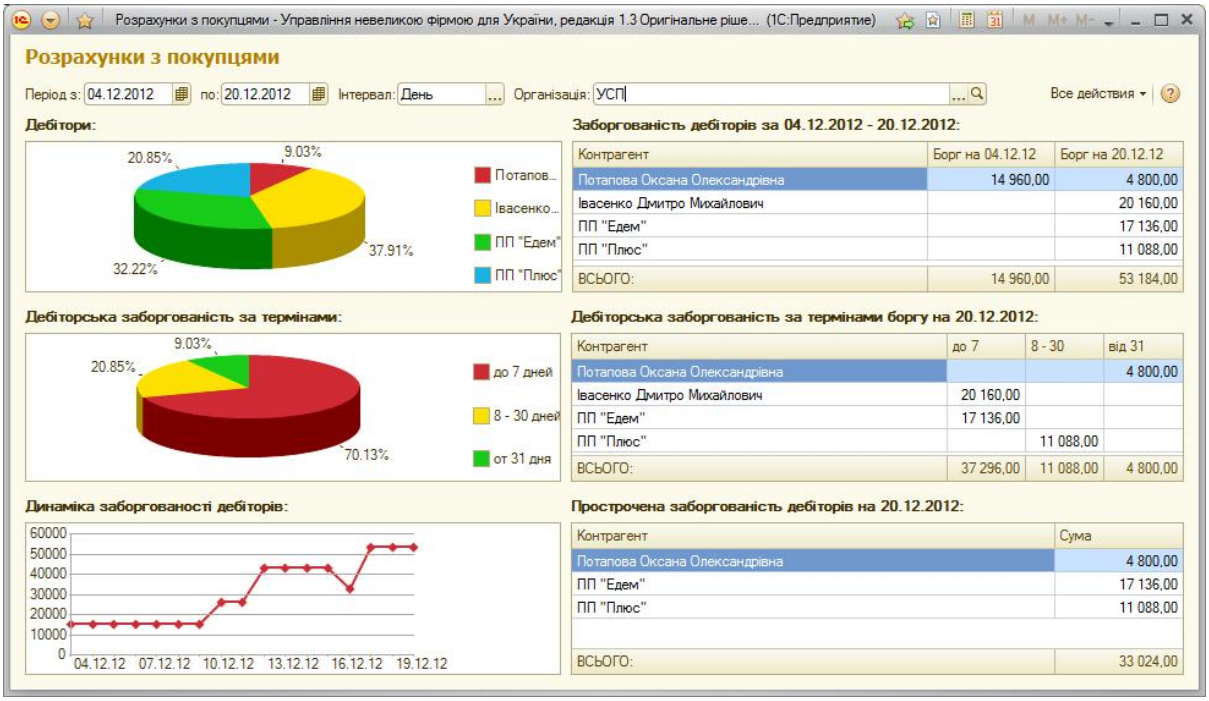

Рис. 1.3. 1С: Підприємство 8

• Оформлення замовлень на матеріали, комплектуючі, інструменти на основі не автоматизованої праці аналітиків.

Огляд функцій модуля управління запасами в системі SAP (рисунок 1.4):

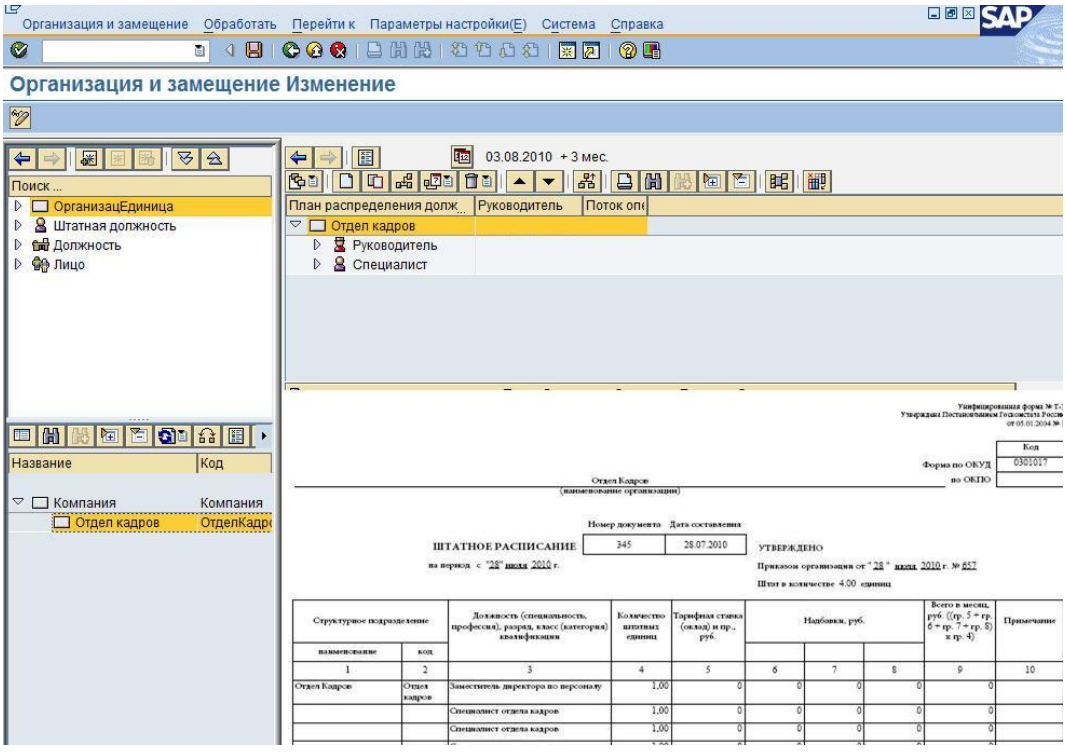

Рис. 1.4. Система SAP

• Облік запасів на складах і базах; методи оцінки, змінна середня і стандартна ціна;

• Управління номерами серій і партій;

• Звіти про асортимент, операції та оцінці запасів;

• Складський і бухгалтерський облік надходжень, розходу і переміщення товарів між складами;

• Комплектування і упаковка для відвантаження;

• Створення та підтримка «багатоетапних» специфікацій виробів;

• Відпуск матеріалів і прийом готових виробів вручну або автоматично;

• Облік і контроль матеріалів на підставі їх кількості. Функції управління складськими запасами відносяться до здійснення внутрішніх складських маніпуляцій і зберігання.

Огляд функцій модуля управління запасами в системі Галактика (рисунок 1.5).

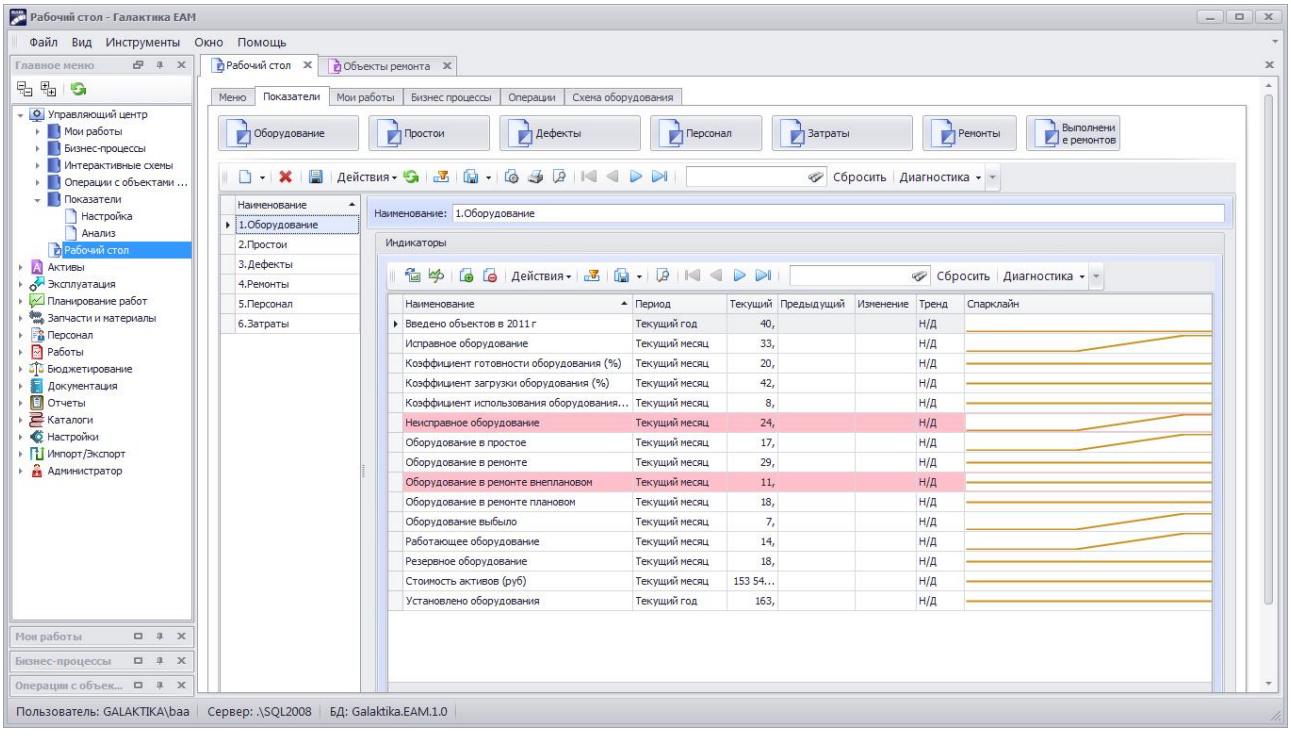

Рис. 1.5. Система Галактика

• Складський оперативний облік матеріалами, комплектуючими, інструменти по різних аналітиках;

• Управління прайс-листами постачальників;

• Ведення ордерного складського обліку;

• Управління партіями і серійними номерами;

• Інвентаризація складу;

Огляд функцій модуля управління запасами в системі Oracle (рисунок 1.6):

• Автоматичне формування оптимального списку постачальників;

• Поповнення запасів: Точка замовлення, Методика 2-х корзин, мінімум-максимум;

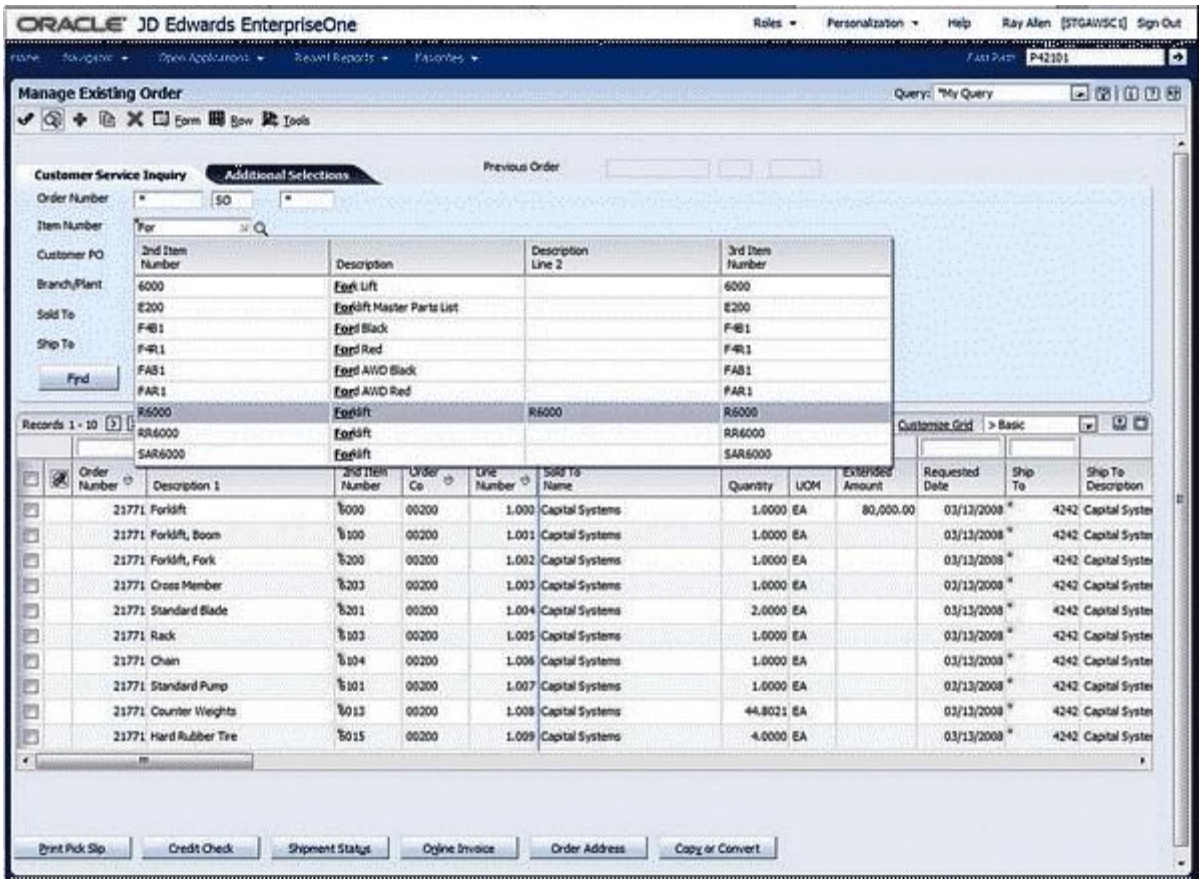

Рис. 1.6. Система Oracle

• АВС аналіз (аналіз груп запасів);

• Управління партіями і серійними номерами;

• Пересортування товару;

• Упаковка та перепакування товару;

• Списання товару;

• Робота з надлишками і недостачами;

• Інвентаризація;

Огляд функцій модуля управління запасами в системі MS Axapta (рисунок 1.7)

• Моніторинг та контроль товарно-матеріальних потоків;

• Отримання актуальної інформації про роботу складу та поточного рівня запасів;

• Інвентаризація складу;

• Перерахунок собівартості по одній з п'яти моделей;

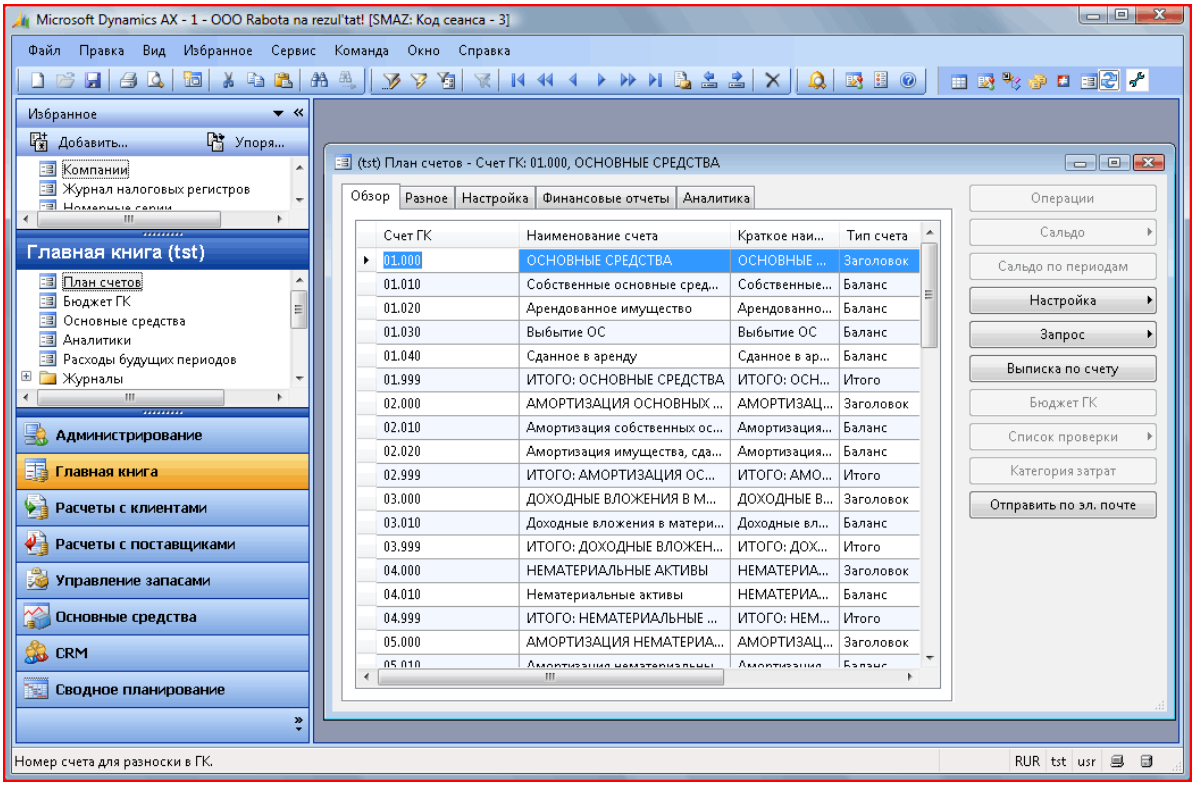

Рис. 1.7. Система MS Axapta

• Ведення партійного обліку, генерація серійних номерів і контроль термінів придатності;

• Робота зі специфікаціями;

• Підтримка необмеженого числа рівнів вкладеності специфікацій;

• Повний комплект графічних засобів для створення специфікації і роботи з ними;

• Підтримка всіх версій однієї специфікації;

• Побудова системи управління запасами для розподілених структур;

Огляд функцій модуля управління запасами в системі RS-Balance 3

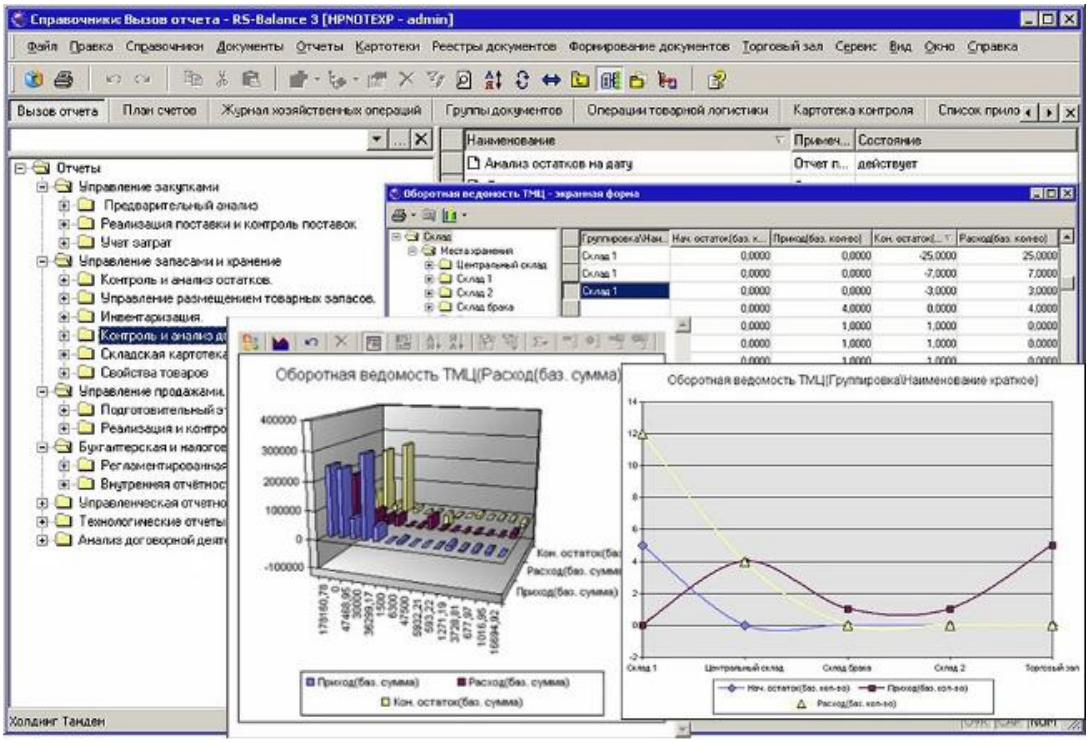

Рис. 1.8. Система RS-Balance 3

• Управління розміщенням і зберіганням матеріальних цінностей;

• Контроль надходження і розходу матеріальних цінностей;

• Організація і проведення інвентаризації, переоцінки;

• Виконання комплектування та розукомплектування;

• Лімітування складських запасів;

• Здійснювати контроль за термінами зберігання і відстежувати дати їх закінчення;

• Вести серійні номери товарів;

• Встановлювати і вести моніторинг гарантійних термінів для контролю повернення від клієнтів або повернення постачальнику неякісних товарів;

• Отримувати інформацію про залишки по партіях на складах і в підрозділах

В ході огляду модулів управління запасами в різних ERP-системах були виділені наступні функції і проведено порівняльний аналіз (таблиця 1.1.).

Таблиця 1.1

Порівняльна характеристика систем управління запасами

Функції	1С	SAP	Галак-тика	Oracle	MS Axapta	RS-Balance 3
Складський та бухгалтерський облік надходжень, переміщення товарів між складами	+	+	+	+	+	+
Облік накладних витрат	+	+	+	+	+	+
Управління номерами серій і партій	+	+	+	+	+	+
Оформлення замовлень на матеріали, комплектуючі, інструменти на основі не автоматизованої праці аналітиків	+	+	+	+	+	+
Методики поповнення запасів	ні	ні	ні	+	ні	ні
АВС аналіз (аналіз груп запасів)	+	ні	ні	+	ні	ні
Іінвентаризація складу	+	+	+	+	+	+
Ведення статистики по браку виробництва	+	+	ні	ні	ні	+
Розрахунок потреб з отриманих розкладів в системах планування	ні	ні	ні	ні	ні	ні
Розрахунок найбільш відповідного числа поставок на заданому періоді для планування поставок за потребами, отриманих на основі розкладів	ні	ні	ні	ні	ні	ні
Розрахунок запасних матеріалів, комплектуючих на основі наявної статистики по моделі управління запасними частинами	ні	ні	ні	ні	ні	ні
Формування замовлень на закупівлю матеріалів, комплектуючих, інструментів на основі наявних прас листів постачальників і на основі розрахованих потреб і кількості поставок.	ні	ні	ні	ні	ні	ні

В існуючих програмних продуктах відсутні можливості управління запасами і матеріально-технічним забезпечення підприємства на основі побудованих розкладів. У зв’язку з цим можливість управління запасами на основі розкладів є безумовно актуальною, цікавою і важливою задачею для машинобудівних підприємств.

1.4. Постановка задачі

В даний час у зв'язку з наявністю потужної обчислювальної техніки з'явилася можливість будувати розклади для обробки деталей і складання виробів з перебуванням найбільш прийнятних варіантів послідовності виконання даних робіт.

Для роботи в безперебійному режимі підприємству необхідно мати потрібні матеріали, комплектуючі вироби до початку виготовлення або зборки того чи іншого об'єкта виробництва. Складність ситуації посилюється тим, що відсутні як моделі, так і програмні продукти з функціями розрахунку потреб за розкладами і розрахунку по них найбільш підходящого числа поставок. Тому існує явна необхідність розробки системи управління запасами на основі побудованих розкладів.

Розглянемо інформацію, яка міститься в розкладах, яку необхідно використовувати для розробки сситеми. Розклади обробки деталей на верстатах будуються для заданого періоду розрахунку деталей для всього обробного обладнання (рисунок 1.9). Розклади містять послідовності деталей, які найбільш підходять для обробки деталей з найменшими витратами часу.

Аналогічно розкладом обробки деталей на верстатах, будуються розклади для збірки виробів. В даний час розробляються інформаційні системи планування, які дозволяють отримувати дані розкладу, як приклад наводяться кафедральні розробки в цій галузі. На рис.1.10 зображено побудований розклад для обробки деталей на верстатах.

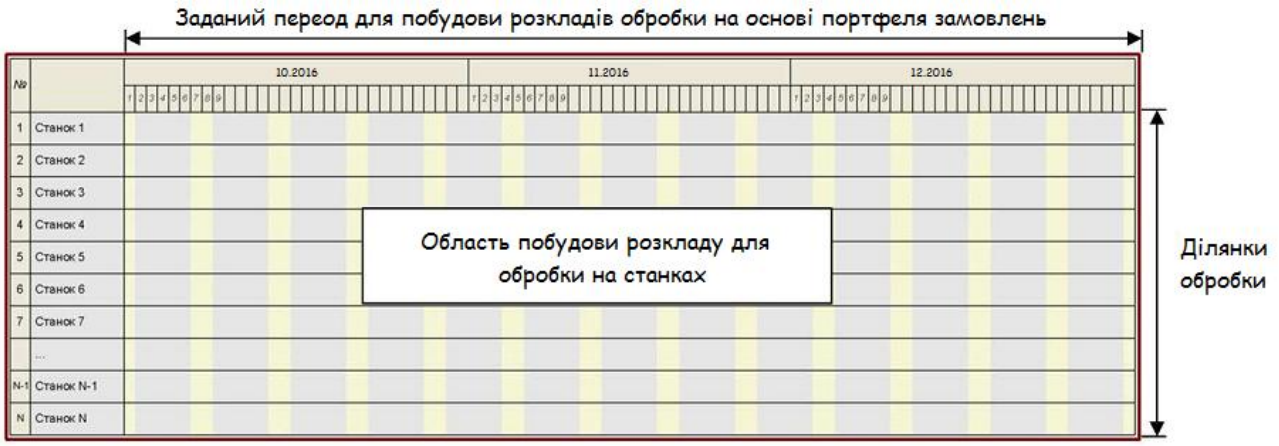

Рис. 1.9. «Форма побудови розкладів для обробки деталей на верстатах»

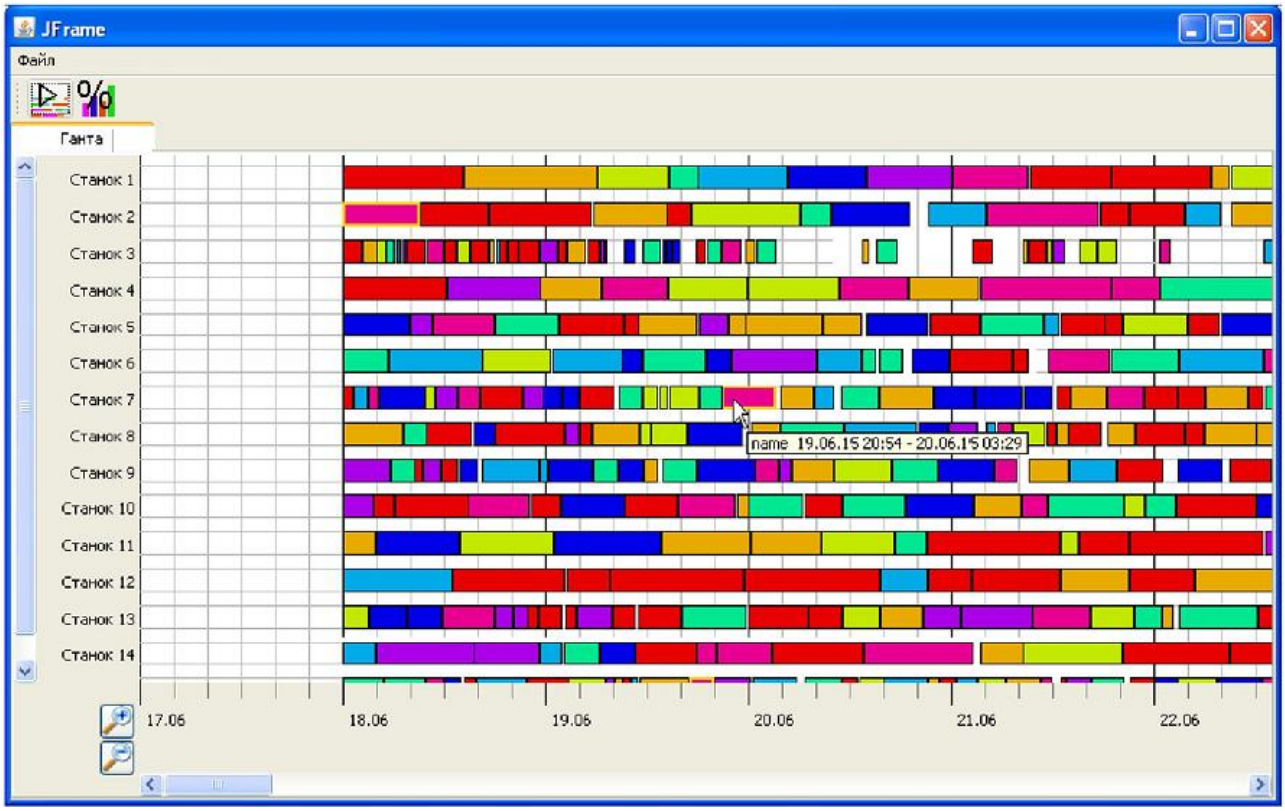

Рис.1.10. «Приклад побудови розкладу для обробки на верстатах в системі планування »

У вказаному прикладі виробництва є замовлення з переліком виробів. Для кожного виробу є ієрархія до рівня потрібних матеріалів, комплектуючих з зазначенням норм витрат на матеріали і комплектуючі.

В ході дипломного проекту потрібно розробити концепцію автоматизації управління запасами матеріалів і комплектуючих виробів для машинобудівних підприємств. Як приклад машинобудівного підприємства розглянути ДП "Автоскладальний завод №1" ПАТ "Автомобільна компанія "Богдан.

На основі розробленої концепції необхідно спроектувати і розробити відкриту систему управління запасами, інтегруються з іншими інформаційними системами підприємства.

Для розробки системи управління запасами необхідно:

• Розробити алгоритм розрахунку запасних деталей, заготовок, комплектуючих на основі моделі управління запасними частинами;

• Розробити алгоритм формування замовлень на поставку матеріалів і комплектуючих на основі порівняння цін прайс-листів постачальників, введених в систему;

Розробити архітектуру системи, що включає в себе підсистему експорту-імпорту даних, клієнтську частину і підсистему звітів у вигляді web-порталу для підрозділів дирекції машинобудівного підприємства.

1.5. Специфікація вимог до системи

Автоматизована система управління запасами для машинобудівних підприємств призначена для автоматизації процесу своєчасного і повного задоволення потреб виробництва в матеріалах, запасних частинах, інструментах і т.д. при мінімальних витратах на їх доставку та зберігання на основі розкладів.

Підсистема управління запасними матеріалами, комплектуючими, деталями, заготовками, складальними одиницями призначена для:

• На основі даних портфеля і статистики по засобах під час виробництва виробів, деталей, заготовок розраховувати необхідну кількість запасних виробів, деталей, заготовок;

• На основі статистичних даних по виходу з ладу інструментів, верстатів розраховувати необхідну кількість запасних інструментів, запасних частин для верстатів;

• Збір та управління статистикою засобів на підприємстві та використання зібраної статистики в розрахунках запасних частин в підсистемі;

• Контроль наявності запасних частин для проведення ремонтних робіт, необхідних для зменшення простоїв виробництва, пов'язаних з поломками обладнання.

Підсистема управління запасами матеріалів та комплектуючими виробами на основі розкладів призначена для:

• На основі розкладів, побудованих в системі планування виробництва на основі маршрутів обробки, норми часу та інших даних, розраховувати потреби в матеріалах і комплектуючих по днях на заданому часовому періоді розкладу;

• Збір та управління статистикою затримок поставок в розрізі постачальників на підприємстві і використання зібраної статистики в розподілі поставок між різними постачальниками;

• Розподіл обсягу поставок між різними постачальниками відповідно до розрахованих потреб за розкладом і автоматизоване формування замовлень по кожному постачальнику у відповідності до найбільш підходящих чисел поставок, розрахованих в системі;

Підсистема управління замовленнями призначена для:

• Ведення складського обліку;

• Ведення замовлень (постачання і збут);

• Контроль виконання замовлень;

Інформаційний портал системи управління запасами призначений для отримання звітів керівництвом підприємства для прийняття управлінських рішень.

• Статистика по засобах у виробництві;

• Звіт по розрахованим запасним частинах

• Звіт по найкращому числу поставок, на даному розрахунковому періоді

• Звіт по побудованих замовленнях на поставку матеріалів, комплектуючих в розрізі постачальників;

• Контроль балансу матеріалів, комплектуючих відповідно до розрахованих потребам;

• Статистика затримок при поставках в розрізі постачальників;

• Оперативний облік в системі;

Метою створення системи є підвищення автоматизації виробництва за рахунок:

- Збільшення ефективності планування виробництва;

- Зменшення простоїв обладнання через дефіцит запасів;

- Виявлення і використання прихованих потужностей виробництва;

- Збільшення ефективності формування плану закупівель;

- Зниження накладних витрат на обробку інформації;

- Збільшення ефективності контролю стану складу;

- Автоматичне формування замовлень;

- Автоматизований розрахунок найбільш потрібної кількості поставок;

Створення єдиного інформаційного простору:

- Забезпечення однозначності і цілісності інформації;

- Забезпечення інформаційної безпеки;

- Усунення дублювання і неоднозначності даних;

- Збільшення ефективності управління виробництвом;

- Підвищення рентабельності капіталу і збільшення прибутку;

- Підвищення інтеграції існуючих процесів;

- Розширення управління знаннями та збільшення аналітичних можливостей;

- Підвищення конкурентоспроможності за рахунок інформаційної складової;

- Зниження вартості задіяних в автоматизації процесів;

- Зниження невиробничих витрат;

Дана система повинна дозволити:

- Поліпшити терміни поставки і виконання замовлень на 5-10%;

- Знизити трудомісткість планування поставки матеріалів на 30-50%;

- Зменшити складські запаси на 20-30%;

Система повинна забезпечувати можливість розрахунку оптимальних величин запасів на підставі обраного модуля системи і моделі управління запасами машинобудівного підприємства

Автоматизована система управління запасами повинна складатися з наступних модулів:

• Підсистема управління запасними матеріалами, комплектуючими, деталями, заготовками, складальними одиницями;

• Підсистема управління запасами матеріалами та комплектуючими виробами на основі розкладів;

• Підсистема управління замовленнями і скадським обліком.

Система повинна мати можливість виведення звіту.

Система повинна забезпечувати можливість спільної роботи користувачів і забезпечувати їх індивідуальними інтерфейсами відповідно до набору прав.

Підсистема управління запасними матеріалами, комплектуючими, деталями, заготовками, складальними одиницями повинна забезпечувати можливість розрахунку запасних частин на основі ймовірностей засобів введених в систему, вартості та штрафів, передбачених в моделі управління запасними частинами;

Підсистема управління запасами матеріалами та комплектуючими виробами на основі розкладів повинна забезпечувати можливість розрахунку найбільш підходящого числа поставок матеріалів та комплектуючих, а також розподіл обсягів поставок між різними постачальниками відповідно до розраховани потреб за розкладами.

Укрупнена функціональна модель підсистеми управління запасами представлена у вигляді діаграми варіантів використання (рисунок 1.11). Дана діаграма являє собою графічне представлення взаємодії користувача і комп’ютерної системи. Кожен варіант використання охоплює деяку очевидну

для користувачів функцію системи і вирішує деяку дискретну задачу користувача.

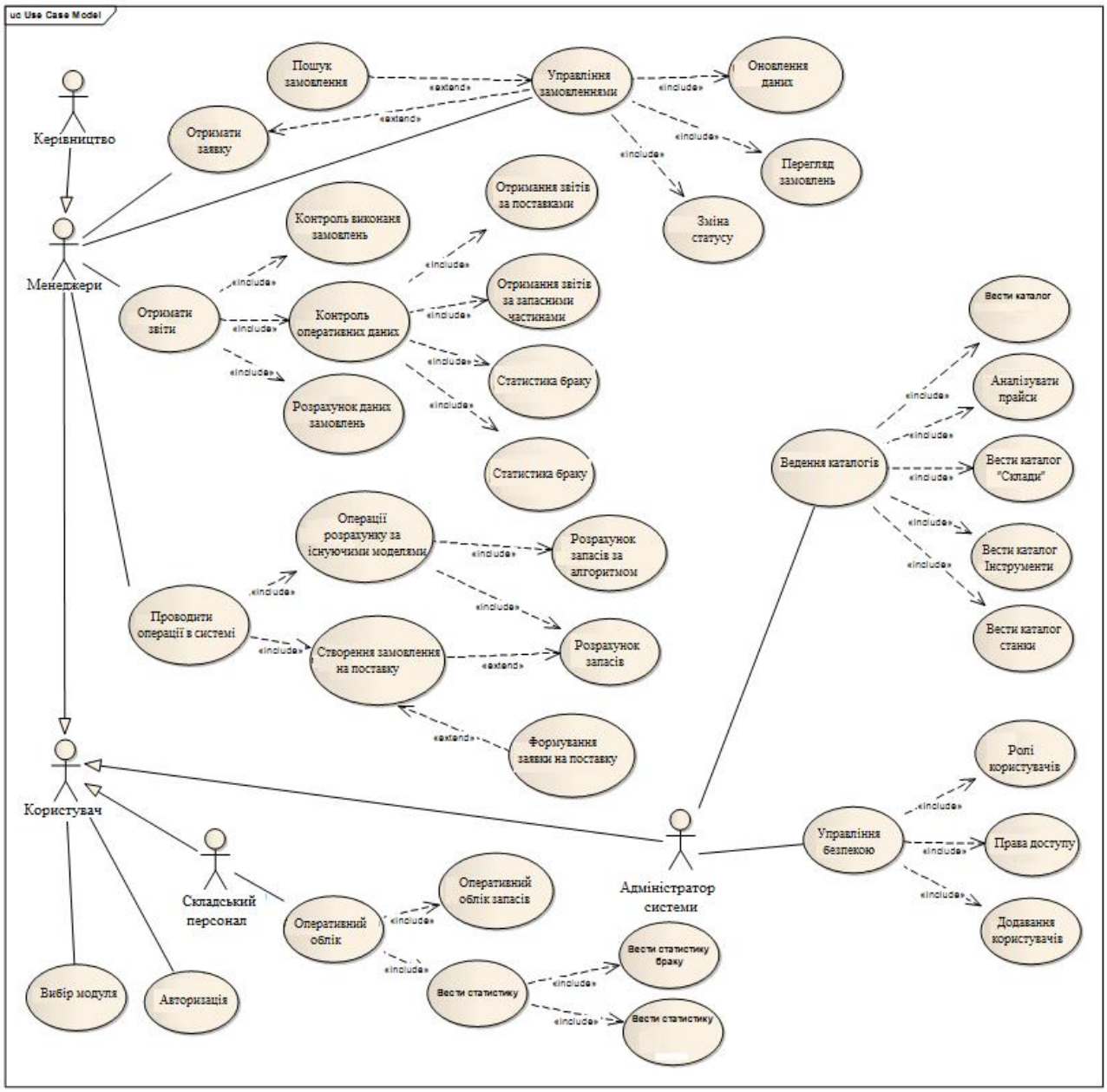

Рис.1.11. Діаграма варіантів використання

Специфікація вимог до програмної системи – це специфікація окремого програмного продукту, програми або набору програм, які виконують деякі дії в деякому середовищі. Тобто – це повний опис поведінки системи що розробляється [3].

В загальному випадку специфікація включає наступне:

- глосарій проекту;
- опис варіантів використання.

Наведемо список основних термінів та понять в області розробки програмної системи «Управління запасами на підприємстві» – глосарій. Глосарій – список понять в специфічній області знання з їх визначеннями [3]. Ці поняття та визначення подано у таблиці 1.2.

Таблиця 1.2

Глосарій

Термін	Опис терміну
1. Основні поняття та категорії предметної області та проекту	
Працівник	особа, яка бере участь у певному трудовому процесі, працює за певним фахом
Запаси на підприємстві	активи підприємства, які: утримуються для подальшого продажу за умов звичайної діяльності; перебувають у процесі виробництва з метою подальшого продажу продукту виробництва;
Брак	продукція, передавання якої споживачу не допускається через наявність дефектів
Організаційна структура управління	упорядкована сукупність служб, відділів, підрозділів і окремих посадових осіб, що знаходяться у взаємозв’язку і співпідпорядкованості і виконують певні управлінські функції [3].
Оперативний облік	спосіб спостереження, відображення та контролю за окремими господарськими та технічними операціями безпосередньо в процесі їх здійснення з метою оператив-ного управління.
Програмне забезпечення	сукупність програм системи обробки інформації і програмних документів, необхідних для експлуатації цих програм [3].
Програмний продукт	програмний засіб, програмне забезпечення, які призначені для постачання користувачів (покупцеві, замовникові) [3].
Бізнес-процес	будь-яка діяльність, що має вхідний продукт, додає вартість до нього, та забезпечує вихідний продукт для внутрішнього або зовнішнього споживача [3].
База даних	логічно впорядкований та взаємопов’язаний набір даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів.

Продовження таблиці 1.2

2. Користувачі системи	
Адміністратор	Особа, яка займається безпосереднім управлінням системою на технічному рівні.
Адміністратор БД	Особа, яка займається безпосереднім управлінням БД.
Керівництво	Особи, які управляють підприємство в цілому і можуть впливати на всіх працівників
Менеджер	Особа, яка здійснює управлінські функції відділу та забезпечує взаємодію відділу з іншими структурними підрозділами.
Складський персонал	Особа, яка здійснює оперативйни облік на складах підприємства
3. Вхідні та вихідні документи	
База даних	логічно впорядкований та взаємопов’язаний набір даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів.
Замовлення	типова форма первинного обліку підприємства
Прайс-лист	документ, в якому визначено специфікацію товарів, їх характеристики та ціни
Розклад	документ, в якому розписано потреби матеріалів за певними часовими характеристиками
Звіт	документ, який забезпечує відображення даних за певною ознакою

У таблицях 1.3 – 1.14 наведено опис варіантів використання, що реалізують основну функціональність програмної системи.

Таблиця 1.3

Варіант використання «Отримати заявку»

Контекст використання	Управління заявками.
Дійові особи	Менеджери
Передумова	Користувач аутентифікований та авторизований.
Тригер	Натиснення кнопки «Створити заявку».
Сценарій	1. Заповнення полів. 2. Перевірка правильності введених даних. 3. Натиснення кнопки «Зберегти».
Постумова	Відображення інформації про заявки.

Таблиця 1.4

Варіант використання «Управління замовленнями»

Контекст використання	Управління замовленнями.
Дійові особи	Всі користувачі відповідно до прав доступу
Передумова	Користувач аутентифікований та авторизований.
Тригер	Вибір замовлення.
Сценарій	-
Постумова	Відображення інформації про замовлення.

Таблиця 1.5

Варіант використання «Перегляд замовлень»

Контекст використання	Управління замовленнями.
Дійові особи	Менеджери
Передумова	Користувач аутентифікований та авторизований.
Тригер	Вибір посади зі списку.
Сценарій	1. Вибір замовлення. 2. Визначення пареметрів. 3. Натиснення кнопки «Переглянути».
Постумова	Відображення списку замовлень

Таблиця 1.6

Варіант використання «Вести статистику на складі»

Контекст використання	Управління інформацією.
Дійові особи	Складський песронал
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «статистика».
Сценарій	-
Постумова	Відображення статистики в розрізі ознак.

Таблиця 1.7

Варіант використання «Оперативний облік запасів на складі»

Контекст використання	Управління складськими запасами.
Дійові особи	Складський песронал
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Облік запасів на складі».
Сценарій	1. Вибір розділу з матеріалами. 2. Заповнення необхідних полів. 3. Перевірка введених даних. 4. Натиснення кнопки «Зберегти».
Постумова	Відображення даних.

Таблиця 1.8

Варіант використання «Управління безпекою»

Контекст використання	Управління безпекою.
Дійові особи	Адміністратор
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Адміністрування».
Сценарій	1. Вибір ролей користувачів. 2. Формування прав доступу. 3. Натиснення кнопки «Зберегти».
Постумова	Відображення параметрів безпеки.

Таблиця 1.9

Варіант використання «Ведення каталогів»

Контекст використання	Управління каталогами.
Дійові особи	Адміністратор
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Каталоги».
Сценарій	1. Вибір відповідного каталоги. 2. Натиснення кнопки «Управління каталогами». 3. Підтвердження операції.
Постумова	Відображення списку каталогів.

Таблиця 1.10

Варіант використання «Реєстрація користувача»

Контекст використання	Управління користувачами.
Дійові особи	Адміністратор.
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Користувачі».
Сценарій	1. Вибір фотокарточки користувача. 2. Заповнення необхідних полів. 3. Перевірка введених даних. 4. Натиснення кнопки «Зберегти».
Постумова	Відображення списку користувачів.

Таблиця 1.11

Варіант використання «Редагування інформації про користувача»

Контекст використання	Управління користувачами.
Дійові особи	Адміністратор.
Передумова	Користувач аутентифікований та авторизований.

Тригер	Відкриття розділу «Користувачі».
Сценарій	1. Вибір користувача. 2. Натиснення кнопки «Редагувати». 3. Заповнення необхідних полів. 4. Перевірка введених даних. 5. Натиснення кнопки «Зберегти».
Постумова	Відображення списку користувачів.

Таблиця 1.12

Варіант використання «Видалення користувача»

Контекст використання	Управління користувачами.
Дійові особи	Адміністратор.
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Користувачі».
Сценарій	1. Вибір користувача. 2. Натиснення кнопки «Видалити». 3. Підтвердження операції видалення.
Постумова	Відображення списку користувачів.

В таблиці 1.13 наведено специфікацію функціональних вимог програмної системи.

Таблиця 1.13

Специфікація функціональних вимог

Іденти-фікатор вимоги	Назва вимоги	Атрибути вимоги		
		Пріоритет	Складність	Контакт
1	Отримати заявку	рекомендоване	середня	Менеджери
2	Управління замовленнями	обов’язкове	висока	Користувачі
3	Перегляд замовлень	обов’язкове	висока	Менеджери
4	Вести статистику на складі	обов’язкове	висока	Складський персонал
5	Оперативний облік запасів на складі	обов’язкове	висока	Складський персонал
6	Управління безпекою	рекомендоване	середня	Адміністратор
7	Ведення каталогів	обов’язкове	висока	Адміністратор
8	Реєстрація користувача	обов’язкове	висока	Адміністратор
9	Редагування інформації про користувача	обов’язкове	висока	Адміністратор
10	Видалення користувача	обов’язкове	висока	Адміністратор

Специфікацію нефункціональних вимог наведено в таблиці 1.14.

Наведемо специфікацію суттєвих для проекту нефункціональних вимог:

1. Застосовність:
   - мінімальний час для навчання звичайних і досвідчених користувачів;
   - відповідність стандартам графічного інтерфейсу.
2. Надійність:
   - постійна безвідмовна робота;
   - пропускна здатність каналу зв’язку 100 Mb/s;
   - забезпечення можливості віддаленого доступу до комп’ютера, на якому буде встановлена система;
   - доступність – 5%.
3. Робочі характеристики:
   - швидкість завантаження інтернет-ресурсу: 0,1 – 1 с;
   - число транзакцій: 100 / 1 с;
   - використання ресурсів: від 1 Gb, в залежності від кількості студентів.
4. Проектні обмеження:
   - Операційна система Microsoft Windows 7/8;
   - MySQL;
   - Eclipse IDE for Java EE Developers;
5. Вимоги до документації
   - наявність інтерактивної довідки.
6. Інтерфейси:
   -інтерфейс користувача – Java-інтерфейс.

Таблиця 1.14

Специфікація нефункціональних вимог

Ідентифікатор вимоги	Назва вимоги	Атрибути вимог		
		Пріоритет	Складність	Контакт
Застосовність				
1.1	Час, необхідний для навчання звичайних і досвідчених користувачів	Рекомендована	Низька	Адміністратор
1.2	Основні вимоги застосовності нової системи відносно інших систем, які знають користувачі	Опційна	Низька	Адміністратор
1.3	Вимоги по відповідальності стандартам графічного інтерфейсу користувача	Рекомендована	Низька	Адміністратор
Надійність				
2.1	Доступність	Обов’язкова	Середня	Адміністратор
2.2	Середній час безвідмовної роботи	Рекомендована	Середня	Адміністратор
2.3	Точність	Обов’язкова	Середня	Адміністратор
Робочі характеристики				
3.1	Використання ресурсів	Рекомендована	Середня	Адміністратор
4.1	Вимоги до технології програмування	Рекомендована	Середня	Адміністратор

Висновки до розділу 1

Здійснено опис предметної області, напрями діяльності. Визначено склад функцій, що входять до бізнес-процесу на основі яких розроблено схему управління бізнес-процесом. Проведено аналіз відомих програмних систем управління запасами на підприємстві. Здійснено аналіз вимог до програмної системи.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ЗАПАСАМИ НА ПІДПРИЄМСТВІ

2.1. Розроблення архітектури програмної системи

В основі широкого розповсюдження комп'ютерних мереж лежить відома ідея поділу ресурсів. Висока пропускна здатність локальних мереж забезпечує ефективний доступ з одного вузла комп'ютерної мережі до ресурсів, які знаходяться в інших вузлах.

Робоча станція призначена для безпосередньої роботи користувача або категорії користувачів і володіє ресурсами, які відповідні локальним потребам даного користувача.

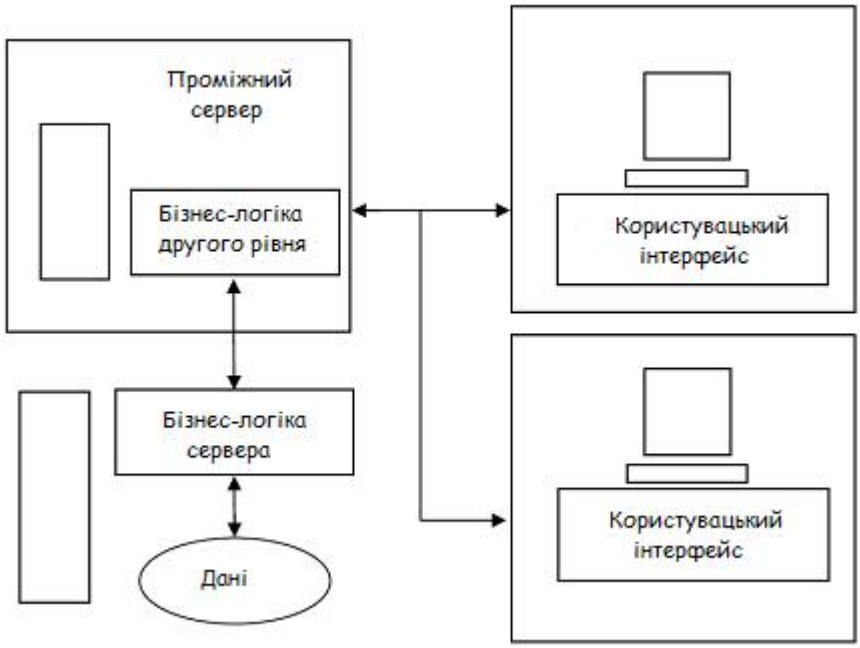

Рис. 2.1. «Трирівнева модель архітектури клієнт/сервер»

Сервер комп'ютерної мережі повинен володіти ресурсами, відповідними його функціональному призначенню і потребам мережі. Стосовно систем баз даних архітектура "клієнт-сервер" цікава і актуальна головним чином тому, що забезпечує просте і відносно дешеве рішення проблеми колективного доступу до баз даних в локальній мережі.

Сервер баз даних - це власне СУБД. Він підтримує всі основні функції СУБД: визначення даних, обробку даних, захист і цілісність даних і т.д. Сервер додатків - це різні додатки, які виконуються "над" СУБД: додатки, написані користувачами, і вбудовані додатки.

Елементами трирівневої архітектури є:

• розподілена БД, яка складається з таблиць локальних БД, які знаходяться на одному вузлі;

• програми доступу до даних і частина прикладних програм, які знаходяться на іншому вузлі (можливо на сервері додатків);

• клієнтські програми на клієнтських вузлах (можливо тільки зовнішній інтерфейс).

Сервер розподіленої БД (Distributed DataBase Server) - це операційна система, яка вирішує наступні завдання:

• управління іменами в розподіленому середовищі (глобальний словник даних);

• оптимізація розподілених запитів;

• управління розподіленими транзакціями.

Доступ до бази даних від прикладної програми або користувача виробляється шляхом звернення до клієнтської частини системи. В якості основного інтерфейсу між клієнтською і серверною частинами виступає мова баз даних SQL.

Структура системи управління запасами реалізована на основі багаторівневої архітектури і приведена на рисунку 2.2.

На сучасному підприємстві використовується багато інформаційних систем автоматизації бізнес-процесів. Для здійснення роботи програми необхідна інтеграція з системою планування, в якій ведеться побудова розкладів. Розроблена система має функції збору різних видів інформації, при наявності інших робочих систем, можна здійснювати імпорт даних в систему, щоб не вести дубльоване введення інформації. Взаємодію систем наведено на рисунку 2.3.

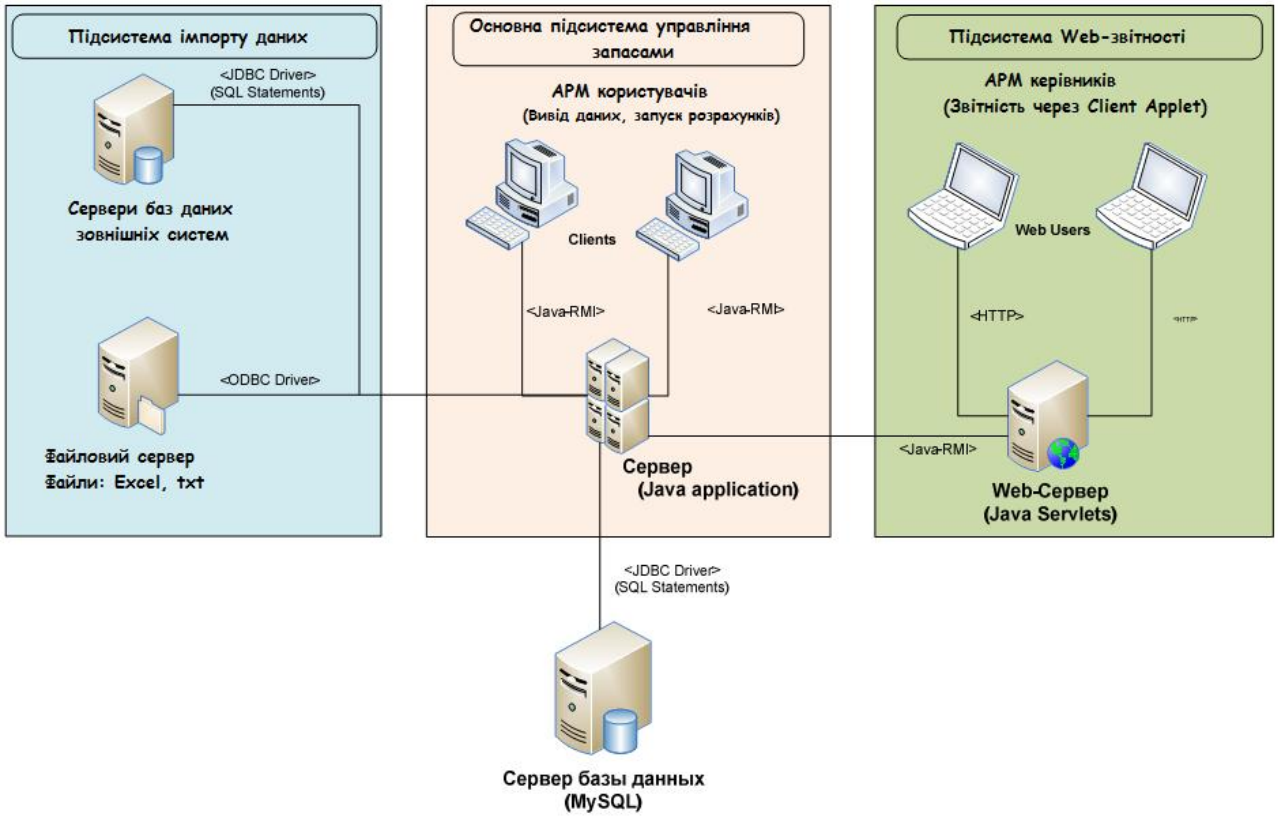

Рис. 2.2. Архітектура системи управління запасами

Рис. 2.3. «Взаємодія системи управління запасами з іншими інформаційними системами»

Алгоритм формування потреб в матеріалах і комплектуючих по розкладах наведено на рисунку 2.4.

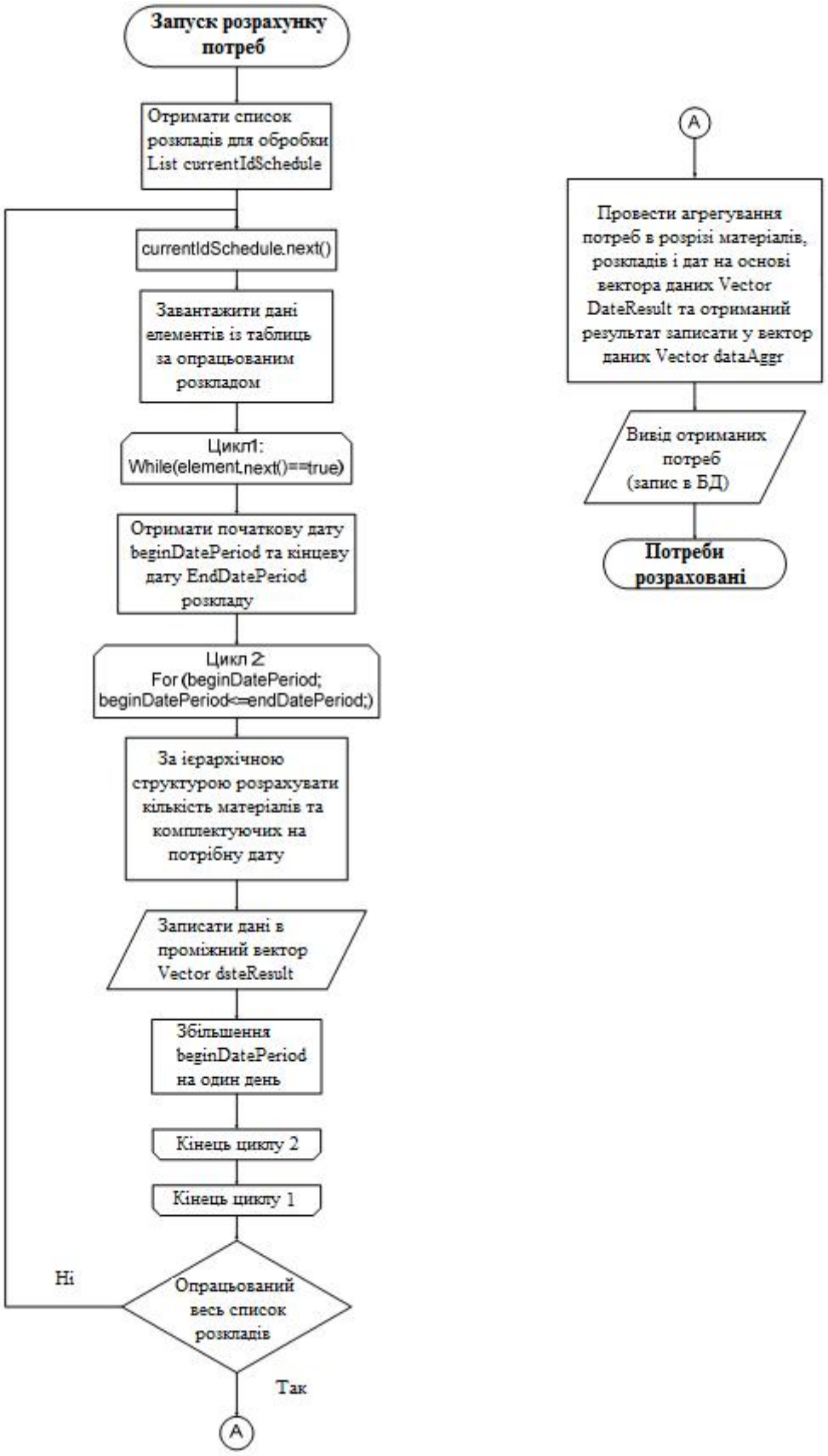

Рис. 2.4. «Алгоритм формування потреб в комплектуючих за розкладами»

Обмеження на операцію формування замовлень з урахуванням наявності складських залишків:

1. Засоби виробництва або засоби матеріалу не розглядається;

2. Період між поставками більше, ніж період замовлення і поставки, що забезпечує відсутність в моделі страхових запасів;

3. Вважається, що розклад розраховується з урахуванням всіх не вироблених деталей, виробів і інших об'єктів виробництва, це забезпечує розрахунок актуальни потреб до моменту замовлення і перешкоджає виникненню ситуацій простою виробництва;

4. Приймається, що число зовнішніх постачальників одно від 2 до 3, при цьому у кожного постачальника представлена вся номенклатура матеріалів і комплектуючих.

2.2. Проектування структури бази даних

В якості моделі даних для проектованої системи була обрана реляційна модель. Виходячи з обраної моделі даних, була спроектована за допомогою CASE-засобів Erwin схема логічної моделі даних (діаграма ERD – модель сутність-зв'язок).

В результаті проведених досліджень предметної області було виявлено, що для роботи системи управління запасами на рівні підприємства необхідна наявність наступних сутностей, представлених в таблицях 2.1-2.43.

Таблиця 2.1

Сутність «Авторизація» та її атрибути

Атрибут	Опис
Логін	Логін користувача
Пароль	Пароль користувача

Таблиця 2.2

Сутність «Зовнішній контрагент» та її атрибути

Атрибут	Опис
id_контрагента	ID код контрагента
Найменування організації	Найменування зовнішньої організації

Таблиця 2.3

Сутність «Допоміжний виробничий інвентар» та її атрибути

Атрибут	Опис
id_інстр	ID код пристосування або інструменту
id_складу	ID код складу, на якому зберігається інвентар
Найменування	Найменування виробничого інвентарю
Маркування	Технічна маркування інвентарю
Залишок	Баланс на складі інвентарю

Таблиця 2.4

Сутність ««Групи користувачів» та її атрибути

Атрибут	Опис
id_групи	ID код Групи користувачів в системі
Найменування групи	Найменування групи користувачів в системі

Таблиця 2.5

Сутність ««Групи прас-листів» та її атрибути

Атрибут	Опис
id_групи прайс-листів	ID код Групи прайс-листів
Найменування групи	Найменування групи

Таблиця 2.6

Сутність ««Групи елементів» та її атрибути

Атрибут	Опис
id_групи	ID код Групи користувачів в системі
Найменування групи	Найменування групи елементів

Таблиця 2.7

Сутність «Дані користувача» та її атрибути

Атрибут	Опис
id Користувача	ID код користувача в системі
Логін	Логін, що належить даному користувачеві
id відділу	ID код відділу, в якому працює користувач
id посади	ID код посади, займаної користувачем
Прізвище	Прізвище користувача
Ім'я	Ім'я користувача
По батькові	По батькові користувача

Таблиця 2.8

Сутність ««Посада» та її атрибути

Атрибут	Опис
id_посади	ID код посади
Найменування посади	Найменування посади користувача

Таблиця 2.9

Сутність ««Доступ до модулів» та її атрибути

Атрибут	Опис
id_групи користувачів	ID код Групи користувачів в системі
id_модуль	ID код модуля

Таблиця 2.10

Сутність «Журнал зміни статусів» та її атрибути

Атрибут	Опис
id записи	ID код запису зміни статусу
id статусу	ID код статусу
id Користувача	ID код Користувача, який змінив статус
Дата	Дата зміни
Попередній статус	ID код попереднього статусу
Новий статус	ID код встановленого статусу

Таблиця 2.11

Сутність «Журнал статистики» та її атрибути

Атрибут	Опис
id статистики	ID код статистики
id типу статистики	ID код типу статистики
id елемента	ID код елемента, за яким вводиться статистика
id працівника	ID код робітника, відповідального за статистикою
Дата запису	Дата запису статистики
Величина	величина статистики

Таблиця 2.12

Сутність «Замовлення» та її атрибути

Атрибут	Опис
id_замовлення	ID код замовлення, введеного в систему
id типу_ замовлення	ID код типу замовлення
id контрагента	ID код зовнішньої організації
id статусу	ID код статусу документа
Порядковий номер замовлення	Порядковий номер замовлення
дата початку	Дата початку дії замовлення
дата закінчення	Дата закінчення дії замовлення

Таблиця 2.13

Сутність «Ієрархія елементів» та її атрибути

Атрибут	Опис
id_parent	ID код елемента батька
id child	ID код елемента нащадка
кількість	Кількість в ієрархії

Таблиця 2.14

Сутність «Оперативний облік допоміжного інвентару» та її атрибути

Атрибут	Опис
id_інструмент	ID код пристосування або інструменту
id оперативного документу	ID код оперативного документа
Кількість інвентар	кількість інвентарю
id_партії	ID код партії елемента

Таблиця 2.15

Сутність «Оперативний облік елементів» та її атрибути

Атрибут	Опис
id_елементу	ID код елементу
id оперативного документу	ID код оперативного документа
Кількість елементів	кількість елементів
id_партії	ID код партії елемента

Таблиця 2.16

Сутність «Оперативний документ» та її атрибути

Атрибут	Опис
id Опер документа	ID код Оперативного документа
id тип документа	ID код типу документа
id контрагента	ID код контрагента
Номер документа	Номер оперативного документа
статус документа	ID код статусу документа
Дата введення	Дата введення документа
дата операції	Дата виробленої операції

Таблиця 2.17

Сутність «Статус документа» та її атрибути

Атрибут	Опис
id_статусу	ID код статусу документа
Найменування статусу документа	Найменування статусу документа

Таблиця 2.18

Сутність ««Відділ» та її атрибути

Атрибут	Опис
id_відділу	ID код відідлу
Найменування відділу	Найменування відділу організації

Таблиця 2.19

Сутність «Партія» та її атрибути

Атрибут	Опис
id партії	ID код партії
Номер партії	Номер партії
Найменування партії	Найменування партії

Таблиця 2.20

Сутність «Користувач-група» та її атрибути

Атрибут	Опис
id_групи	ID код групи
Логін	Логін користувача

Таблиця 2.21

Сутність «Прайс-лист об’єктів виробництва» та її атрибути

Атрибут	Опис
id_елемента	ID код елемента
id_контрагента	ID код контрагента
Дата введення	Дата введення прайс-листа
початок дії	Початок дії прайс-листа
закінчення дії	Закінчення дії прайс-листа
id_группи	ID код групи прайс-листів

Таблиця 2.22

Сутність «Програмний модуль» та її атрибути

Атрибут	Опис
id_модуль	ID код модуля
Найменування модуля	Найменування модуля

Таблиця 2.23

Сутність «Виробнича ділянка» та її атрибути

Атрибут	Опис
id_ділянки	ID код ділянки
Найменування ділянки	Найменування ділянки виробництва

Таблиця 2.24

Сутність «Робоча зміна» та її атрибути

Атрибут	Опис
id робочої зміни	ID код робочої зміни
час початку	Час початку роботи виробничої зміни
час закінчення	Час закінчення роботи виробничої зміни

Таблиця 2.25

Сутність «Працівник» та її атрибути

Атрибут	Опис
id робочого	ID код працівника
id робочої зміни	ID код працівника зміни
Прізвище	Прізвище працівника
Ім'я	Ім'я працівника
По батькові	По батькові працівника

Таблиця 2.26

Сутність «Працівник-ділянка» та її атрибути

Атрибут	Опис
id працівника	ID код працівника
id виробничої дільниці	ID код виробничої дільниці

Таблиця 2.27

Сутність «Склад» та її атрибути

Атрибут	Опис
id_складу	ID код складу
Найменування складу	Найменування складу

Таблиця 2.28

Сутність «Склад прас-листа» та її атрибути

Атрибут	Опис
id Прайс-листа	ID код прайс
id елемента	ID код елемента прайсу
Ціна	Вартість елемента в прайсі

Таблиця 2.29

Сутність «Специфікація замовлення» та її атрибути

Атрибут	Опис
id замовлення	ID код замовлення
Id_елемента	ID код елемента замовлення
Кількість виробів	Кількість елементів в замовлені

Таблиця 2.30

Сутність «Статистика постачань» та її атрибути

Атрибут	Опис
id статистики	ID код статистики
id контрагента	ID код зовнішньої організації
Дата статистики	Дата введення статистики
К днів затримки	Кількість днів затримок

Таблиця 2.31

Сутність «Статуси замовлень» та її атрибути

Атрибут	Опис
id статусу	ID код статусу замовлень
статус	Найменування статусу

Таблиця 2.32

Сутність «Тип замовлення» та її атрибути

Атрибут	Опис
Id_типу замовлення	ID код типу замовлення
Тип замовлення	Тип замовлення

Таблиця 2.33

Сутність «Тип оперативного документу» та її атрибути

Атрибут	Опис
id типу документа	ID код типу документа
Найменування типу	Найменування типу документа
знак операції	Знак операції при операції приходу

Таблиця 2.34

Сутність «Тип статистики елементів» та її атрибути

Атрибут	Опис
id типу статистики	ID код типу статистики
Найменування типу статистики	

Таблиця 2.35

Сутність «Тип елементу» та її атрибути

Атрибут	Опис
Id_типу елементу	ID код
Найменування типу елементу	Найменування типу елементу

Таблиця 2.36

Сутність «Елемент специфікації» та її атрибути

Атрибут	Опис
id_Елемента	ID код елемента
id_тип елемента	ID код типу елемента
id_складу	ID код складу, на якому зберігається об'єкт
id_групи	ID код групи елементів
Найменування	Найменування елементів
маркування	маркування елемента
залишок елементів	Баланс елемента на складі

Таблиця 2.37

Сутність «Розклад» та її атрибути

Атрибут	Опис
id_розкладу	ID код розкладу
id елемента	ID код елемента, що використовується в розкладі
id обладнання	ID код обладнання, на якому відбувається обробка елементу
id типу розкладу	ID код типу розкладу
Час початку	Час початку
Час закінчення	Час закінчення

Таблиця 2.38

Сутність «Потреба у матеріалах» та її атрибути

Атрибут	Опис
id_розкладу	ID код розкладу
id елемента	ID код елемента, по якому розраховується потреба
Дата	Дата розрахунку
Величина потреби	Величина потреби

Таблиця 2.39

Сутність «Обладнання» та її атрибути

Атрибут	Опис
id_обладнання	ID код обладнання
Найменування обладнання	Найменування обладнання
Маркування обладнання	Маркування обладнання
id_ділянки	ID код виробничої дільниці, на якій встановлено обладнання

Таблиця 2.40

Сутність «Розрахунок» та її атрибути

Атрибут	Опис
id_розрахунку	ID код розрахунку
Початок	початок розрахунку
Кінець	закінчення розрахунку
Початок періоду розрахунку	Початок періоду розрахунку кількості поставок
Кінець періоду розрахунку	Кінець періоду розрахунку кількості поставок
id_розкладу	ID код розкладу, за яким здійснюється розрахунок
Затримки	Вартість витрат на оформлення замовлення

Таблиця 2.41

Сутність «Результати розрахунку» та її атрибути

Атрибут	Опис
id_розрахунку	ID код розрахунку
id_результату	ID код результату
id_Типу величини	ID код типу величини
Значення	розраховане значення

Таблиця 2.42

Сутність «Тип розкладу» та її атрибути

Атрибут	Опис
Id_типу розкладу	ID код
Найменування типу розкладу	Найменування типу розкладу

Таблиця 2.43

Сутність «Тип розрахункової величини» та її атрибути

Атрибут	Опис
Id_типу величини	ID код типу величини
Найменування величини	Найменування величини

На рисунку 2.5 приведена логічна структура бази даних, що описує основні суттості та їх взаємозв'язок. На підставі цієї інформації згодом розробляється фізична структура БД.

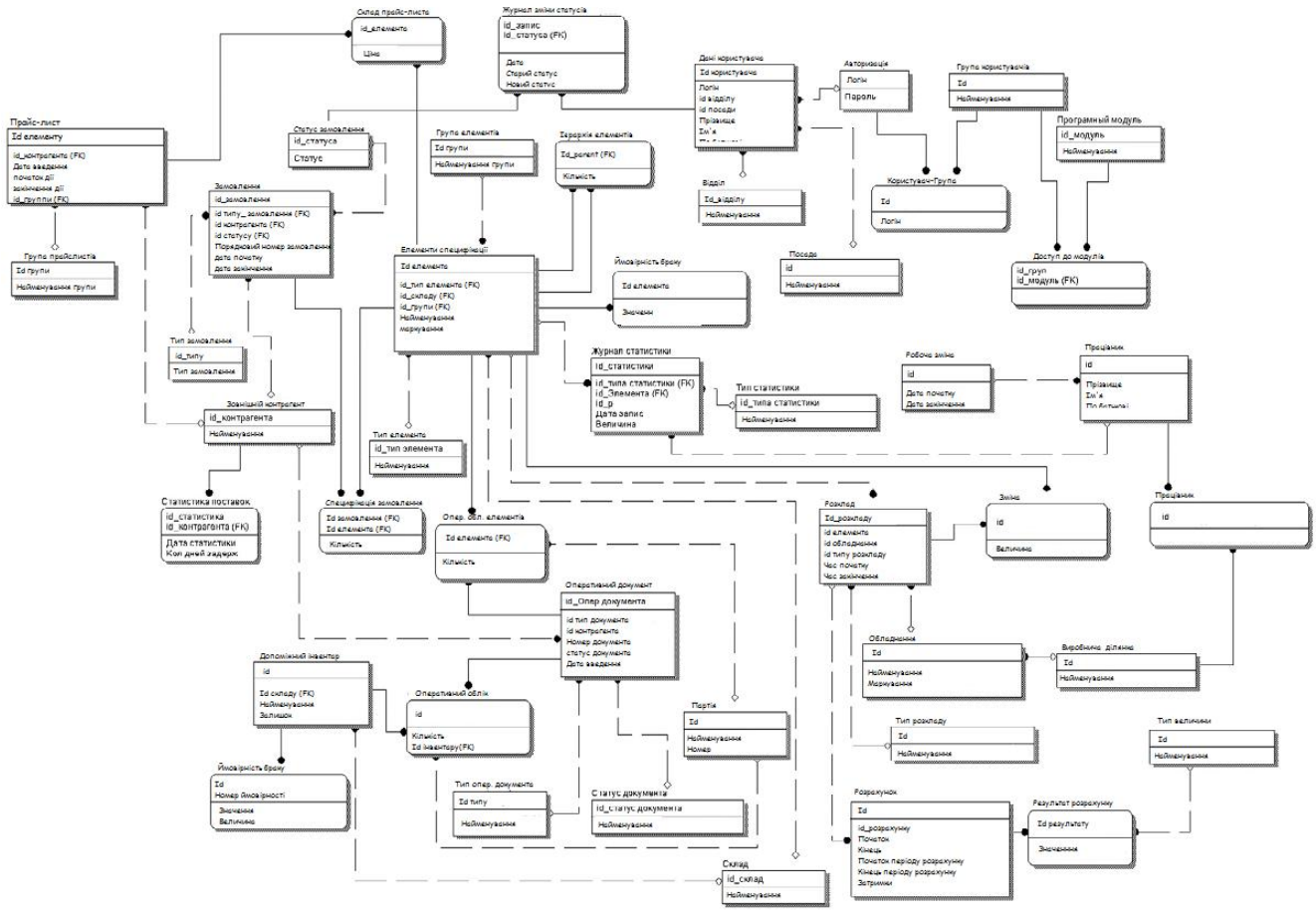

Рис. 2.5. Логічна структура бази даних

Висновки до розділу 2

У цьому розділі розроблено архітектуру програмного додатку, що дозволить краще зрозуміти функції основних його частин. Створено та описано структурну схему, основними компонентами якої є: рівень клієнта, рівень бізнес-логіки та рівень даних. Описано функціональну структуру системи та її основних елементів – модулів обробки даних. Визначено основні елементи бази даних та встановлено зв’язки між ними. Спроектовано структуру бази даних.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ ЗАПАСАМИ НА ПІДПРИЄМСТВІ

3.1. Програмна реалізація системи

Для реалізації системи управління запасами на підприємстві вибраного мову програмування Java. Java є компілюючою мовою програмування, тобто створені нею програми за допомогою компілятора перетворюються у виконуємий файл. Процес такого перетворення складається з двох етапів. Спочатку, код програми на мові Java за допомогою програми-транслятора переводиться у так званий байт-код, який може бути перетворений у машинний код за допомогою віртуальної машини Java –Java Virtual Machine (JVM) (рисунок 3.1). Це дозволяє в подальшому виконувати програму написану на мові програмування Java на комп'ютерах, які працюють під керуванням різноманітних операційних систем. Єдиною вимогу для запуску програми є встановлення на комп'ютері відповідної віртуальної машини Java. Таким чином досягається максимальна мобільність створених програм та їх незалежність від платформи виконання.

Мова програмування Java побудована на принципах об'єктно-орієнтованого програмування. Основою будь-якої програми на мові Java є клас. Кожний клас складається з властивостей та методів. Властивості – це характеристики деякого об'єкту , а методи – це процедури та функції які реалізують виконання дій над об'єктом та його властивостями.

Кожний створений клас уявляю собою тип даних. Для використання в програмі створюються так звані «об'єкти» або екземпляри класу. При створенні кожного нового об'єкту йому надаються властивості, які повинні бути у кожного екземпляру класу, а також методи роботи з ними.

Наприклад, якщо створюється програма для роботи зі списком замовлень, то створюється клас на основі якого формується множина об’єктів, кожний з яких зберігає інформацію про конкретнее замовлення. Властивостями кожного екземпляру класу можуть бути: найменування, номер ідентифікаційний код, шифр підрозділу та інше. Методами такого класу є процедури та функції призначені для створення запису про новее замовлення, видалення замовлення зі списку, отримання шифру, зміни найменування і таке інше.

Взаємодія програми на мові Java з користувачем та зовнішнім середовищем побудована на обробці подій. Подіями можуть бути: натискання користувачем кнопки, виклик пункту меню, зміна візуального компоненту на екранній формі та інші. У разі виникнення події автоматично викликається так званий оброблювач події, який, у свою чергу, запускає на виконання задану підпрограму. У якості підпрограм, що оброблюють події, зазвичай виступають методи головного класу програми.

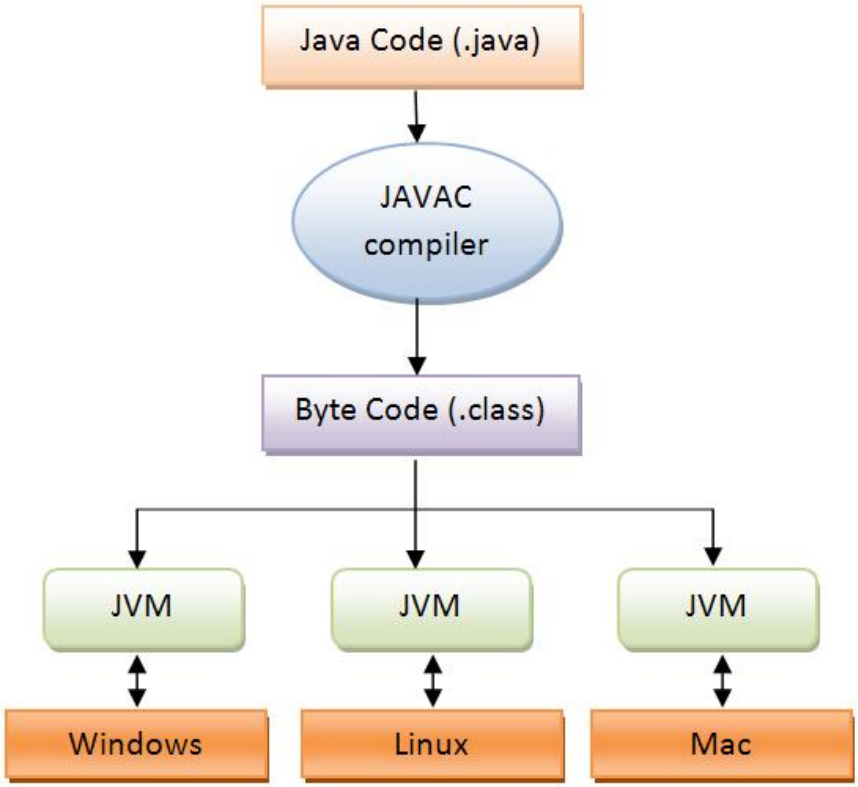

Рис. 3.1. Схема виконання програм на Java

При розробці прототипу системи управління запасами на підприємстві були спроектовані і реалізовані програмні класи, представлені на рисунку 3.2.

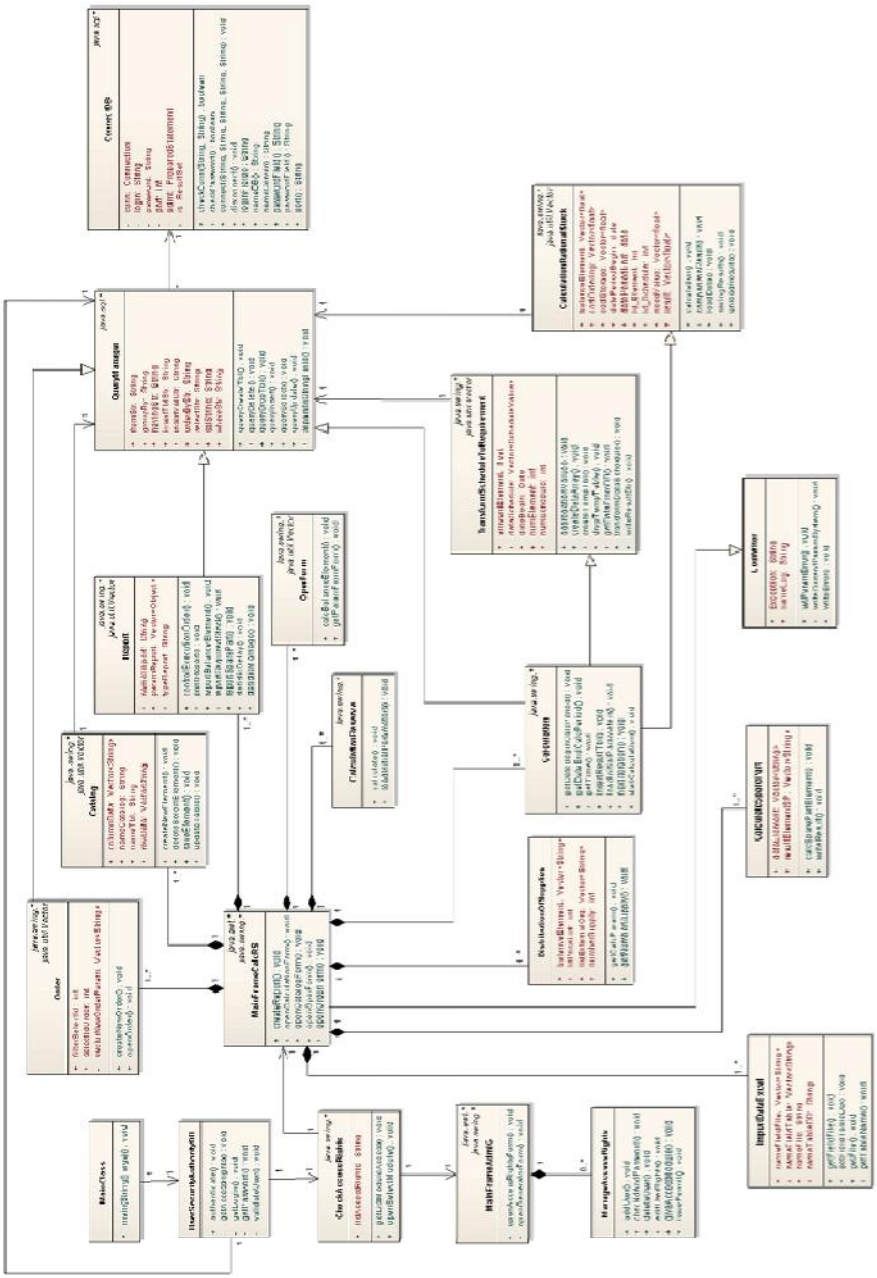

Рис. 3.2. Діаграма класів системи

Розглянемо детальніше програмну реалізацію системи управління запасами на підприємстві. На рисунку 3.3 представлено схему реалізації класу «Employee». Java має явну реалізацію через розширення (extends) одного класу іншим.

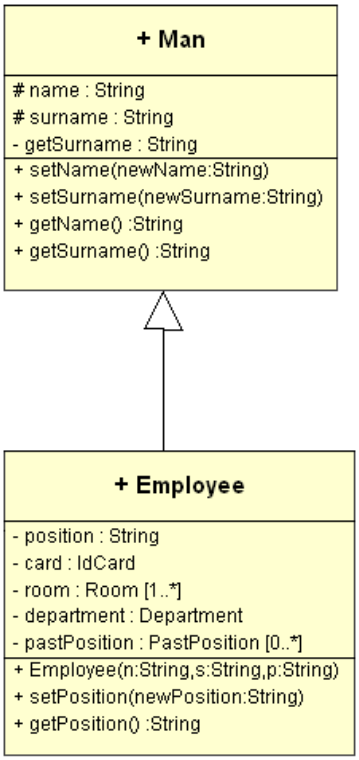

Рис. 3.3. Реалізація класу Employee

Клас «Man» (людина) - більш абстрактний, а «Employee» (співробітник) більш спеціалізований. Клас «Employee» успадковує властивості і методи «Man». Лістинг коду для цієї діаграми буде мати наступний вигляд:

```
public class Man{
protected String name;
    protected String surname;
    public void setName(String newName){
        name = newName;
    }
    public String getName(){
        return name;
    }
    public void setSurname(String newSurname){
        name = newSurname;
    }
    public String getSurname(){
        return surname;
    }
}
// успадковуємо клас Man
public class Employee extends Man{
    private String position;
    // створюємо конструктор
    public Employee(String n, String s, String p){
        name = n;
        surname = s;
```

```
        position = p;
    }
    public void setPosition(String newProfession){
        position = newProfession;
    }
    public String getPosition(){
        return position;
    }
}
```

Введемо в модель клас Department (відділ) – оскільки наше підприємство структуровано по відділах (рисунок 3.4). У кожному відділі може працювати один або більше осіб. Можна сказати, що відділ включає в себе одного або більше співробітників і таким чином їх агрегує. На підприємстві можуть бути співробітники, які не належать ні одному відділу, наприклад, директор підприємства.

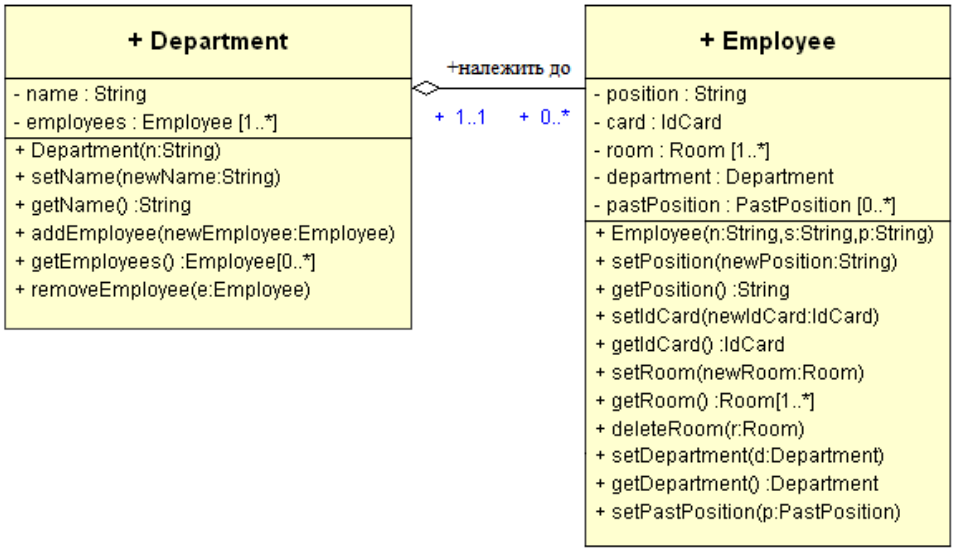

Рис. 3.4. Агрегація класів Department та Employee

Розглянемо реалізацію класу Department:

```
public class Department {
     private String name;
     private Set employees = new HashSet ();
     public Department (String n) {
         name = n;
     }
     public void setName (String newName) {
         name = newName;
     }
     public String getName () {
         return name;
     }
```

```
    public void addEmployee (Employee newEmployee) {
        employees.add (newEmployee);
        // Пов'язуємо співробітника з цим відділом
        newEmployee.setDepartment (this);
    }
    public Set getEmployees () {
        return employees;
    }
    public void removeEmployee (Employee e) {
        employees.remove (e);
    }
}
```

Даний клас, крім конструктора і методу зміни назви відділу, має методи для занесення до відділу нового співробітника, для видалення співробітника і для отримання всіх співробітників, які належать до даного відділу. Навігація на діаграмі не відображена, значить вона є двобічною: від об'єкта типу «Department» можна дізнатися про співробітника і від об'єкта типу «Employee» можна дізнатися до якого відділу він відноситься. Так як нам потрібно легко дізнаватися до якого відділу відноситься будь-якої співробітник, то додамо в клас Employee поле і методи для призначення та одержання відділу:

```
...
private Department department;
...
public void setDepartment(Department d){
    department = d;
}
public Department getDepartment(){
    return department;
}
```

Однією з вимог до нашої системи є вимога про те, щоб зберігати дані про історію займаних посад на підприємстві. Введемо новий клас «pastPosition». У нього, крім властивості «ім'я» (name), введемо і властивість «department», яке зв'яже його з класом «Department» (рисунок 3.5).

Дані про минулі займані посади є частиною даних про співробітника, таким чином між ними зв'язок ціле-частина і в той же час, дані про минулі посади не можуть існувати без об'єкта типу «Employee». Знищення об'єкта «Employee» має призвести до знищення об'єктів «pastPosition».

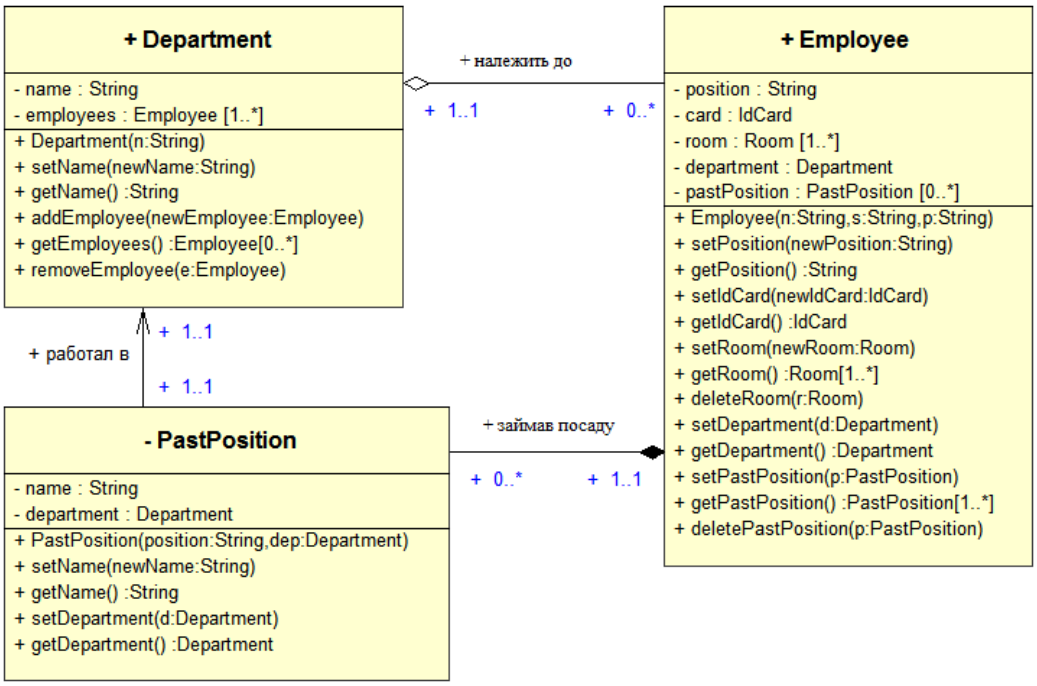

Рис. 3.5. Схема композиції класів системи

Розглянемо лістинг класу «PastPosition»:

```
private class PastPosition{
    private String name;
     private Department department;
     public PastPosition(String position, Department dep){
         name = position;
         department = dep;
     }
     public void setName(String newName){
         name = newName;
     }
     public String getName(){
         return name;
     }
     public void setDepartment(Department d){
         department = d;
     }
     public Department getDepartment(){
         return department;
     }
}
```

В клас Employee додано властивості і методи для роботи з даними про попередні займані посади:

```
...
private Set pastPosition = new HashSet();
...
public void setPastPosition(PastPosition p){
     pastPosition.add(p);
```

```
}
public Set getPastPosition(){
     return pastPosition;
}
public void deletePastPosition(PastPosition p){
     pastPosition.remove(p);
}
...
```

Зауважимо, що клас «Menu» не відноситься до прикладної області, а являє собою «системний» клас програми. Розглянемо лістинг даного класу:

```
public class Menu {
     private static int i = 0;
     public static void showEmployees (Employee [] employees) {
         System.out.println ("Список співробітників:");
         for (i = 0; i <employees.length; i ++) {
             if (employees [i] instanceof Employee) {
                  System.out.println (employees [i] .getName () + "-" + employees [i]
.getPosition ());
             }
         }
     }
}
```

Реалізація, як і наслідування має явне вираження в мові Java: оголошення інтерфейсу і можливість його реалізації деяким класом.

Для демонстрації відношення «реалізація» створимо інтерфейс «Unit». Досліджуване підприємство ділитися не тільки на відділи, але і на групи, ділянки. Інтерфейс «Unit» являє собою саму абстрактну одиницю поділу. У кожній одиниці відділу є свої замовлення товарів, тому метод для отримання кількості замовлень буде актуальний для кожного класу реалізовує інтерфейс «Unit».

Інтерфейс Unit:

```
public interface Unit{
     int getPersonCount();
}
```

Реалізація в класі «Department»:

```
public class Department implements Unit{
    ...
    public int getPersonCount(){
        return getEmployees().size();
    }
```

Лістинг основних модулів системи представлено в додатку А.

3.2. Програмна реалізація бази даних

Фізична модель розробляється для СУБД - MySQL Server 5.0. Розробка фізичної моделі даних грунтувалася на побудованої логічної моделі даних. Для розробки фізичної моделі даних використовувалося програмне забезпечення - MySQL-Front версія 5.0 (Build: 1.234).

Конфігурація MySQL Server:

• Influence memory - Developer Machine;

• Multifunctional Database;

• Approximate number of concurrent connections to the server – Manual Setting (40);

• Port number – 3306;

• Enable TCP / IP Networking;

• Enable Strict Mode;

• Default Character Set - cp1251;

• Service Name – MySQL;

Далі наводиться опис таблиць, полів, індексів таблиць і інших параметрів бази даних system:

Имя	Тип
Primary Index	login

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
login	char(15)				
password	char(21)				

Рис. 3.6. Таблиця authorization

Имя	Тип
Primary Index	Id_calculation

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_calculation	int(11)			auto_increment	
calcBegin	datetime	NULL	NULL		
calcEnd	datetime	NULL	NULL		
beginPeriodCalc	date	NULL	NULL		
endPeriodCalc	date	NULL	NULL		
Id_schedule	int(11)	NULL	NULL		
costOrdering	float(9,2)	NULL	NULL		

Рис. 3.7. Таблиця calculation

Имя	Тип
Primary Index	Id_order, Id_element

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_order	int(11)		'0'		
Id_element	int(11)		'0'		
amountElement	int(11)	NULL	NULL		

Рис. 3.8. Таблиця content_order

Имя	Тип
Primary Index	Id_priceList, Id_element

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_priceList	int(11)		'0'		
Id_element	int(11)		'0'		
price	float(9,2)	NULL	NULL		

Рис. 3.9. Таблиця content_price_list

Имя	Тип
Primary Index	Id_department

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_department	int(11)			auto_increment	
nameDepartment	varchar(255)	NULL	NULL		

Рис. 3.10. Таблиця department

Имя	Тип
Primary Index	Id_element

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_element	int(11)			auto_increment	
Id_typeElement	int(11)	NULL	NULL		
id_warehouse	int(11)	NULL	NULL		
Id_groupElement	int(11)	NULL	NULL		
nameElement	varchar(255)	NULL	NULL		
code	varchar(255)	NULL	NULL		
balanceElement	float(9,3)	NULL	NULL		
probabilityDefect	float(5,3)	NULL	NULL		

Рис. 3.11. Таблиця element_spec

Имя	Тип
Primary Index	Id_equipment

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_equipment	int(11)			auto_increment	
Id_productionArea	varchar(255)	NULL	NULL		
nameEquipment	varchar(255)	NULL	NULL		
codeEquipment	char(20)	NULL	NULL		

Рис. 3.12. Таблиця equipment

Имя	Тип
Primary Index	Id_externalOrg

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_externalOrg	int(11)			auto_increment	
nameOrganization	varchar(255)	NULL	NULL		

Рис. 3.13. Таблиця external_organization

Имя	Тип
Primary Index	Id_groupElement

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_groupElement	int(11)			auto_increment	
nameGroupElement	varchar(255)	NULL	NULL		

Рис. 3.14. Таблиця group_element

Имя	Тип
Primary Index	Id_groupPriceList

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_groupPriceList	int(11)			auto_increment	
nameGroupPriceList	varchar(255)	NULL	NULL		

Рис. 3.15. Таблиця group_price_list

Имя	Тип
Primary Index	Id_parentElement, Id_childElement

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_parentElement	int(11)		'0'		
Id_childElement	int(11)		'0'		
amount	float(9,3)	NULL	NULL		

Рис. 3.16. Таблиця hierarchy_element

Имя	Тип
Primary Index	Id_incumbency

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_incumbency	int(11)			auto_increment	
nameIncumbency	varchar(255)	NULL	NULL		

Рис. 3.17. Таблиця incumbency

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_journalStatistic	int(11)			auto_increment	
Id_typeStatistic	int(11)	NULL	NULL		
Id_element	int(11)	NULL	NULL		
Id_worker	int(11)	NULL	NULL		
dateRecord	date	NULL	NULL		
Value	float(7,2)	NULL	NULL		

Рис. 3.18. Таблиця journal_statistic_element

Имя	Тип
Primary Index	Id_journalStatus, Id_status, Id_user

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_journalStatus	int(11)		'0'		
Id_status	int(11)		'0'		
Id_user	int(11)		'0'		
dateRecord	datetime	NULL	NULL		
oldStatus	varchar(255)	NULL	NULL		
newStatus	varchar(255)	NULL	NULL		

Рис. 3.19. Таблиця journal_status

Имя	Тип
Primary Index	Id_userGroup, Id_module

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_userGroup	int(11)		'0'		
Id_module	int(11)		'0'		

Рис. 3.20. Таблиця module_access

Имя	Тип
Primary Index	Id_schedule, Id_element, dateNeed

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_schedule	int(11)		'0'		
Id_element	int(11)		'0'		
dateNeed	date		'0000-00-00'		
needValue	float(9,3)	NULL	NULL		

Рис. 3.21. Таблиця need_element

Имя	Тип
Primary Index	Id_element, Id_operDoc

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_element	int(11)		'0'		
Id_operDoc	int(11)		'0'		
Id_party	int(11)	NULL	NULL		
amountElement	float(9,3)	NULL	NULL		

Рис. 3.22. Таблиця oper_account_element

Имя	Тип
Primary Index	Id_instrument, Id_operDoc

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_instrument	int(11)		'0'		
Id_operDoc	int(11)		'0'		
Id_party	int(11)	NULL	NULL		
amountInstrument	int(11)	NULL	NULL		

Рис. 3.23. Таблиця oper_account_instrument

Имя	Тип
Primary Index	Id_operDoc

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_operDoc	int(11)			auto_increment	
Id_typeDoc	int(11)	NULL	NULL		
Id_externalOrg	int(11)	NULL	NULL		
Id_statusDoc	int(11)	NULL	NULL		
numberDoc	char(15)	NULL	NULL		
dateInput	date	NULL	NULL		
dateOperation	date	NULL	NULL		

Рис. 3.24. Таблиця oper_document

Имя	Тип
Primary Index	Id_order

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_order	int(11)			auto_increment	
Id_typeOrder	int(11)	NULL	NULL		
Id_externalOrg	int(11)	NULL	NULL		
Id_status	int(11)	NULL	NULL		
numberOrder	char(15)	NULL	NULL		
dateBegin	date	NULL	NULL		
dateEndOrder	date	NULL	NULL		

Рис. 3.25. Таблиця order

Имя	Тип
Primary Index	Id_party

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_party	int(11)			auto_increment	
numberParty	char(20)	NULL	NULL		
nameParty	varchar(255)	NULL	NULL		

Рис. 3.26. Таблиця party

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_element	int(11)			auto_increment	
Id_externalOrg	int(11)	NULL	NULL		
Id_groupPriceList	int(11)	NULL	NULL		
dateInput	date	NULL	NULL		
dateBegin	date	NULL	NULL		
dateEnd	date	NULL	NULL		

Рис. 3.27. Таблиця price_list_element

Имя	Тип
Primary Index	Id_productionArea, nameProductionArea

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_productionArea	int(11)		'0'		
nameProductionArea	varchar(255)		''		

Рис. 3.28. Таблиця production_area

Имя	Тип
Primary Index	Id_instrument

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_instrument	int(11)			auto_increment	
Id_warehouse	int(11)	NULL	NULL		
nameInstrument	varchar(255)	NULL	NULL		
codeInstrument	varchar(255)	NULL	NULL		
balance	int(11)	NULL	NULL		

Рис. 3.29. Таблиця production_instrument

Имя	Тип
Primary Index	Id_module

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_module	int(11)			auto_increment	
nameModule	varchar(255)	NULL	NULL		

Рис. 3.30. Таблиця program_module

Имя	Тип
Primary Index	Id_calculation, Id_result

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_calculation	int(11)		'0'		
Id_result	int(11)		'0'		
Id_typeValueCalc	int(11)	NULL	NULL		
valueCalc	float(9,2)	NULL	NULL		

Рис. 3.31. Таблиця result_calc

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_schedule	int(11)			auto_increment	
Id_element	int(11)	NULL	NULL		
Id_equipment	int(11)	NULL	NULL		
Id_typeSchedule	int(11)	NULL	NULL		
timeBegin	datetime	NULL	NULL		
timeEnd	datetime	NULL	NULL		

Рис. 3.32. Таблиця schedule

Имя	Тип
Primary Index	Id_shift

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_shift	int(11)			auto_increment	
timeBegin	datetime	NULL	NULL		
timeEnd	datetime	NULL	NULL		

Рис. 3.33. Таблиця shift

Имя	Тип
Primary Index	Id_statisticSupply, Id_externalOrg

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_statisticSupply	int(11)		'0'		
Id_externalOrg	int(11)		'0'		
dateRecordStatistic	date	NULL	NULL		
valueStatistic	int(11)	NULL	NULL		

Рис. 3.34. Таблиця statistic_supply

Имя	Тип
Primary Index	Id_statusDoc

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_statusDoc	int(11)			auto_increment	
nameStatusDoc	char(25)	NULL	NULL		

Рис. 3.35. Таблиця status_oper_doc

Имя	Тип
Primary Index	Id_status

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_status	int(11)			auto_increment	
nameStatus	varchar(255)	NULL	NULL		

Рис. 3.36. Таблиця status_order

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_typeElement	int(11)			auto_increment	
nameTypeElement	varchar(255)	NULL	NULL		

Рис. 3.37. Таблиця type_element

Имя	Тип
Primary Index	Id_typeDoc

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_typeDoc	int(11)			auto_increment	
nameTypeDoc	varchar(255)	NULL	NULL		
signOperation	char(15)	NULL	NULL		

Рис. 3.38. Таблиця type_oper_doc

Имя	Тип
Primary Index	Id_typeOrder

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_typeOrder	int(11)			auto_increment	
nameTypeOrder	varchar(255)	NULL	NULL		

Рис. 3.39. Таблиця type_order

Имя	Тип
Primary Index	Id_typeSchedule

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_typeSchedule	int(11)			auto_increment	
nameTypeSchedule	varchar(255)	NULL	NULL		

Рис. 3.40. Таблиця type_shedule

Имя	Тип
Primary Index	Id_typeStatisticElement

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_typeStatisticElement	int(11)			auto_increment	
nameTypeStatistic	varchar(255)	NULL	NULL		

Рис. 3.41. Таблиця type_statistic_element

Имя	Тип
Primary Index	Id_typeValueCalc

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_typeValueCalc	int(11)			auto_increment	
nameValueCalc	varchar(255)	NULL	NULL		

Рис. 3.42. Таблиця type_value_calc

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_user	int(11)			auto_increment	
login	char(15)	NULL	NULL		
Id_department	int(11)	NULL	NULL		
Id_incumbency	int(11)	NULL	NULL		
surname	varchar(255)	NULL	NULL		
name	varchar(255)	NULL	NULL		
patronymic	varchar(255)	NULL	NULL		

Рис. 3.43. Таблиця user_data

Имя	Тип
Primary Index	Id_userGroup

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_userGroup	int(11)			auto_increment	
nameGroup	varchar(255)	NULL	NULL		

Рис. 3.44. Таблиця user_group

Имя	Тип
Primary Index	Id_groupUser, login

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_groupUser	int(11)		'0'		
login	char(15)		''		

Рис. 3.45. Таблиця user_group_access

Имя	Тип
Primary Index	Id_warehouse

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_warehouse	int(11)			auto_increment	
nameWarehouse	varchar(255)	NULL	NULL		

Рис. 3.46. Таблиця warehouse

Имя	Тип
Primary Index	Id_worker

Поля:

Имя	Тип	NULL	По умолчанию	Дополнительно	Комментарии
Id_worker	int(11)			auto_increment	
Id_shift	int(11)	NULL	NULL		
surname	varchar(255)	NULL	NULL		
name	varchar(255)	NULL	NULL		
patronymic	varchar(255)	NULL	NULL		

Рис. 3.47. Таблиця worker

Висновки до розділу 3

У даному озділі обґрунтовано технологію, мову програмування та розроблено програмну систему. Обґрунтовано засоби розробки бази даних та створено програмну реалізацію бази даних програмної системи за допомогою механізму збережених процедур.

РОЗДІЛ 4
ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

4.1. Тестування

В процесі реалізації системи управління запасами на підприємстві було проведено процедуру тестування. Розглянемо детальніше процедуру модульного тестування системи, оскільки вона є дуже важливою з точки зору успішності реалізації проекту.

Спочатку запускаємо інтегроване середовище розробки Eclipse (integrated development environment - IDE), створюємо базовий Java-проект, в який імпортуємо JAR-бібліотек JUnit, jMock і RMock. Java-проект - TestingPersonal. На рисунку 4.1 представлено процедуру створення тестового проекту в середовищі Eclipse.

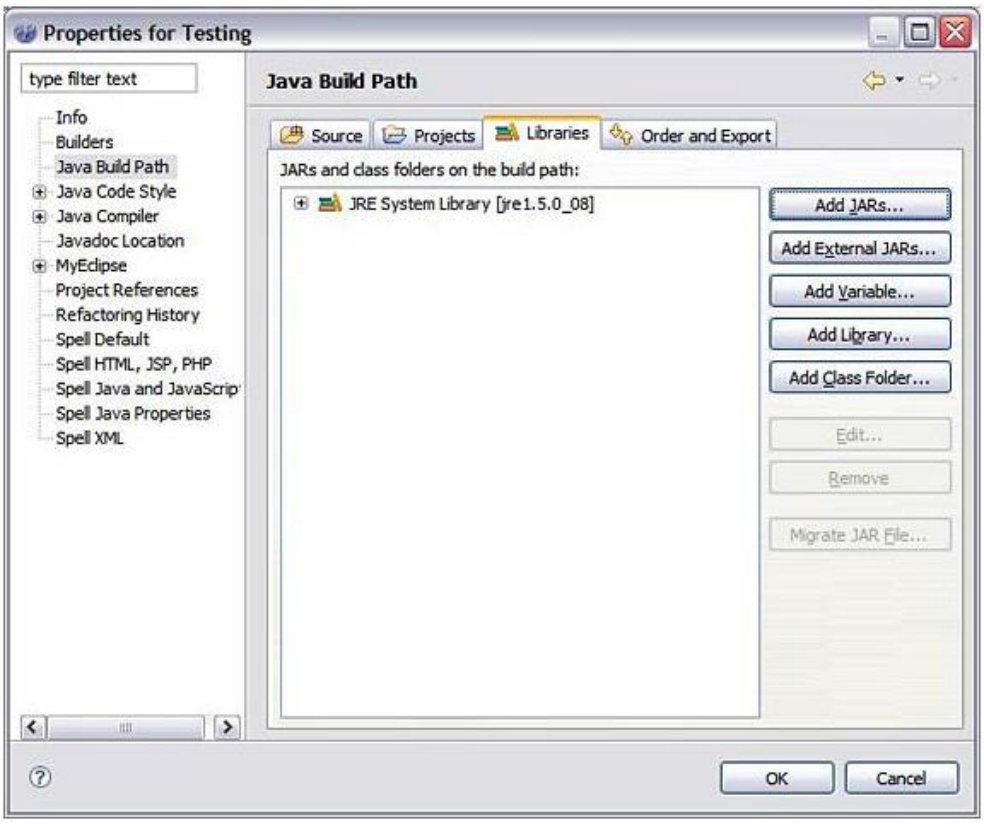

Рис. 4.1. Процедура створення тестового проекту

Використовуємо кнопку Add JARs, якщо JAR-файли вказані в Java classpath (Java Runtime Environment (JRE), налаштована в Eclipse). Кнопка Add Variable працює з конкретним каталогом файлової системи (локальної або віддаленої), де розміщені ресурси (включаючи JAR-файли), на які можна послатися. Використовуємо кнопку Add Library, якщо потрібно послатися на такі спеціалізовані ресурси, які використовуються в Eclipse за замовчуванням або налаштовані на спеціалізоване середовище робочої області Eclipse. Натискаємо кнопку Add Class Folder, щоб додати ресурс з однієї з папок існуючих проектів, уже налаштованих як частину проекту (рисунок 4.2).

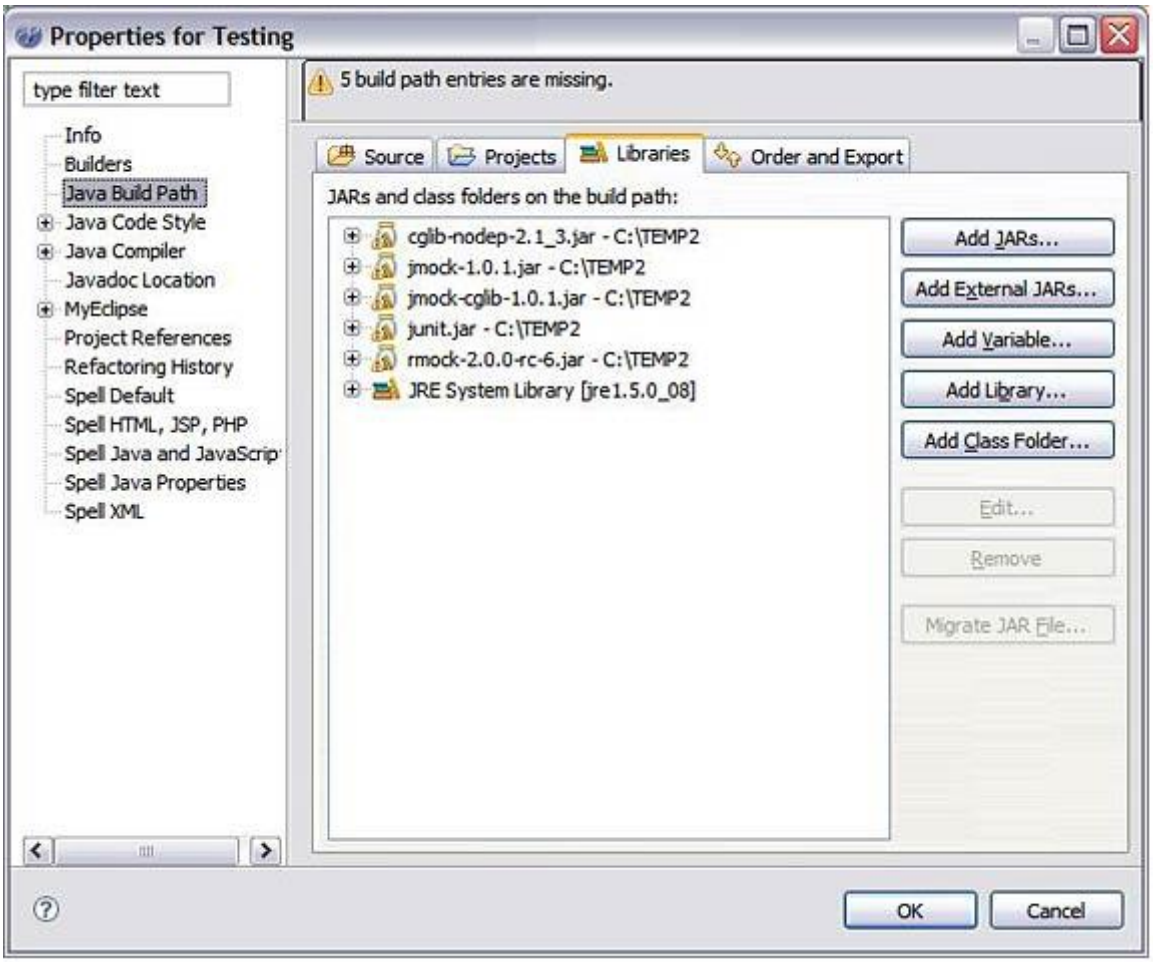

Рис. 4.2. Процедура підключення необхідних бібліотек

Розглянемо детальніше процедуру тестування модуля TestingOrder, а саме з вихідним кодом чотирьох класів: ServiceClass.java, Collaborator.java,

ICollaborator.java, ServiceClassTest.java. Тестованим класом є ServiceClass, який містить один метод: runService (). Метод service приймає об'єкт Collaborator, який реалізує простий інтерфейс ICollaborator. Один метод реалізований в конкретному класі Collaborator:executeJob ().Collaborator. Це клас, який ми повинні імітувати відповідним чином. Четвертий клас - це тестовий клас ServiceClassTest (реалізація максимально спрощена). У лістингу нижче показаний вихідний код цього четвертого класу.

```
public class ServiceClass {
  public ServiceClass(){
  //конструктор без аргументів
  }

  public boolean runService(ICollaborator collaborator){
  if("success".equals(collaborator.executeJob())){
            return true;
  }
  else
  {
            return false;
  }
}
}
```

В класі ServiceClass блок коду if ... else є простим логічним переходом, що допомагає відобразити, чому тест завершиться невдало або успішно при виборі одного (а не іншого) шляху, відповідно до очікуваних результатів. Вихідний код класу Collaborator показаний нижче.

```
public class Collaborator implements ICollaborator{
   public Collaborator(){
  //конструктор без аргументів
   }
   public String executeJob(){
     return "success";
   }
}
```

Клас Collaborator з конструктором без аргументів і простою змінною String, що повертається з методу executeJob(), теж не складний. Нижче показаний код класу ICollaborator.

```
public interface ICollaborator {
     public abstract String executeJob ();
}
```

Інтерфейс ICollaborator має один метод, який повинен бути реалізований в класі Collaborator. Маючи наведений вище код, переходимо до розгляду того, як можна успішно виконати тест класу ServiceClass в різних сценаріях.

Використання jMock для імітації інтерфейсів. Тестуємо метод service в класі ServiceClass. Припустимо, що предметом тестування є твердження, що

метод runService() не виконувався, або, іншими словами, що повернений Boolean-результат дорівнює false. У цьому випадку імітується передача в метод runService() об'єкт ICollaborator для очікування виклику його методу executeJob() і повернення рядка, відмінного від "success". Таким чином гарантуємо, що Boolean-рядок false повертається в тест. Код класу ServiceClassTest, який наведений нижче, містить логіку тесту.

```
import org.jmock.cglib.MockObjectTestCase;
public class ServiceClassTest extends MockObjectTestCase {
        private ServiceClass serviceClass;
        private Mock mockCollaborator;
        private ICollaborator collaborator;
                public void setUp(){
                serviceClass = new ServiceClass();
                mockCollaborator = new Mock(ICollaborator.class);
        }
                public void testRunServiceAndReturnFalse(){
                mockCollaborator.expects(once()).method\
               ("executeJob").will(returnValue("failure"));
                collaborator = (ICollaborator)mockCollaborator.proxy();
                boolean result = serviceClass.runService(collaborator);
                assertFalse(result);
```

Зазвичай гарною ідеєю є включення в тести методу setUp(), якщо в різних прикладах тестів виконуються спільні операції. Метод tearDown() теж годиться, але він не настільки необхідний доти, поки ви не будете виконувати інтегровані тести. В jMock середовище перевіряє всі очікувані результати по всіх фіктивним об'єктах в кінці (або під час) виконання тесту. Немає реальної необхідності включати метод verify() для кожного очікуваного результату. При виконанні тесту як JUnit, тест завершується успішно (рисунок 4.3).

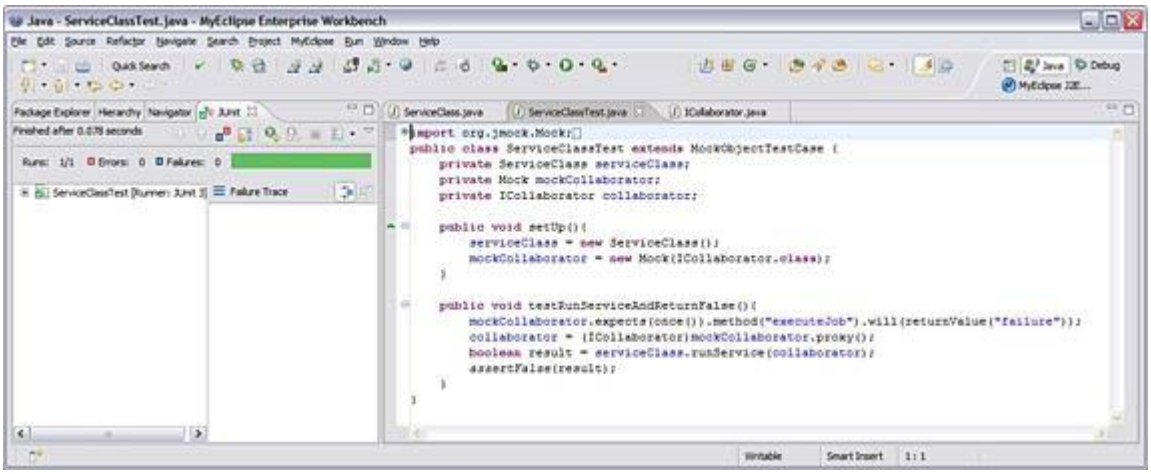

Рис. 4.3. Результати тестування

Клас ServiceTestClass розширює клас org.jmock.cglib.MockObjectTestCase jMock CGLIB. mockCollaborator - це простий клас org.jmock.JMock. Зазвичай є два способи створення фіктивних об'єктів в jMock: для імітації інтерфейсу використовується новий метод Mock (Class.class). Для імітації конкретного класу використовується метод mock (Class.class, "identifier").

Важливо відзначити, як імітованим proxy передається в метод runService() класу ServiceClass. У jMock можна витягти реалізації proxy із створених фіктивних об'єктів, для яких очікувані результати вже були встановлені.

4.2. Розгортання програмного продукту

Для розгортання системи складемо список необхідних вимог до програмного забезпечення:

- Операційна система Microsoft Windows 7/8;
- MySQL;
- Eclipse IDE for Java EE Developers;
- Драйвера JDBC-MySQL;

Вимоги до апаратних ресурсів визначатимуться кількістю користувачів системи. Для невеликої кількості (до 100 одночасно) користувачів достатньо наступної конфігурації для апаратних ресурсів:

- процесор з тактовою частотою не менше 2 Ghz;
- оперативна пам’ять 4 Gb;
- об’єм жорсткого диску має бути не менше 512 Gb; необхідна пропускна здатність залежить від кількості користувачів.

Розглянемо процедуру розгортання проекту, яка складається з двох частин:

- розгортання Java додатку;
- налатування доступу до сервера баз даних MySQL;

Адаптер сервера WTP Eclipse - це інструмент для розгортання та тестування ресурсів Java EE на сервері WebSphere Application Server Community

Edition. Перед тим, як приступити до розгортання ресурсів Java EE, слід визначити новий сервер і середовище виконання сервера. Після створення або імпорту проекту Java EE, в інтегроване середовище розробки (IDE) Eclipse, вказуємо середовище виконання сервера WebSphere Application Server Community Edition в якості цільової середовища. Ця дія додає бібліотеки класів сервера в шлях компонування проекту.

Ресурси, розгорнуті за допомогою функцій Eclipse, рекомендується видаляти і повторно розгортати також за допомогою функцій Eclipse. Наприклад, якщо розгорнути ресурс в Eclipse і потім видалити його за допомогою Web-консолі або команди deploy, то Eclipse буде вважати, що ресурс розгорнутий. Для усунення такої неполадки слід видалити ресурси, які були опубліковані на сервері і вилучені за межами середовища Eclipse.

Для того щоб розгорнути ресурси Java EE на локальному сервері, виконаємо нступні дії:

- У проекції Java EE вибераємо панель Проект і правою кнопкою миші на потрібному проекті Java EE. Вибераємо Виконати як, Запустити на сервері. На панелі Запустити на сервері, якщо сервер встановлено, вибераємо опцію Вибрати існуючий сервер і вибрати потрібний сервер. Якщо сервер Community Edition не заданий, вибераємо опцію Задати новий сервер вручну щоб задати новий сервер. Натискаємо кнопку Готово. Адаптер сервера WTP незабаром розгорне ресурси Java EE. Якщо сервер не запущений, адаптер сервера WTP запустить його і розгорне ресурс Java EE після ініціалізації сервера.

При необхідності ресурс, опублікований на сервері, можна видалити за допомогою опції Додати/Видалити проект. Якщо просто видалити ресурс без видалення пов'язаного проекту, ресурс залишиться розгорнутим на сервері. На панелі Сервер клацаємо правою кнопкою миші на сервері, в якому вимагається розгорнути ресурс. У контекстному меню вибераємо Додати/Видалити проекти.

Процедура розгортання ресурсу Java EE на віддаленому сервері аналогічна розглянутої вище, однак перед її виконанням рекомендується звернути увагу на додаткові особливості.

Для визначення віддаленого сервера необхідно спершу задати локальний сервер і потім в атрибуті імені хоста вказати ім'я хоста віддаленого сервера. Це обов'язкова процедура, оскільки середовище Eclipse використовує бібліотеки класів локального сервера.

За допомогою Eclipse не можна запустити, зупинити або перезапустити віддалений сервер. За допомогою Eclipse не можна запустити віддалений сервер в режимі налагодження. Як правило, внаслідок такого обмеження зручно вибрати підхід, що передбачає розробку і налагодження ресурсів Java EE на локальному сервері з подальшим переходом на віддалений сервер, призначений для інтеграції ресурсів кількох розробників. Відома неполадка Apache Geronimo може викликати непередбачену виняткову ситуацію java.net.UnknownHostExc.

Незалежно від того, яким чином вказано ім'я хоста цільового сервера (навіть у тому випадку, якщо вказано повне ім'я або IP-адреса), цільовий сервер повертає неповне ім'я хоста. Іншими словами він повертає клієнту своє коротке ім'я. Відповідно з цим ім'ям клієнт запускає операцію передачі файлу. Якщо мережа не може перетворити це ім’я, видається повідомлення про виняткову ситуацію. Якщо команда ping дозволяє звернутися до цільової системі за неповним іменем хоста, віддалене розгортання повинно працювати правильним чином. Опис способу обходу цієї неполадки наведено в розділі Усунення неполадок.

Якщо локальна система та цільової сервер розділені брандмауером, необхідно додатково налаштувати передачу запитів HTTP і RMI між ними. Після встановлення сервера за замовчуванням застосовуються порт HTTP 8080 і порт RMI 1099. Якщо в конфігурації сервера вказані інші порти, необхідно додатково налаштувати брандмауер для застосування цих портів. Під час виклику команди deploy повинен виконуватися віддалений сервер. Перед розгортанням або оновленням ресурсу Java EE пов'язані файли передаються по мережі і зберігаються в якості тимчасових файлів.

Налаштування драйвера JDBC-MySQL. Драйвер jdbc треба встановлювати та підключати додатково. Наприклад, для MySQL його можна

взяти на http://dev.mysql.com/downloads/mirror.php?id=412737. Для роботи драйвера JDBC необхідно налаштувати OpenOffice.org/LibreOffice для роботи з Java. Отриманий файл-архів треба розпакувати в каталог. Потім підключити його до OpenOffice.org/LibreOffice. Сервіс-Параметри -OpenOffice.org - Java - Шлях класу-Додати архів і вибрати файл драйвера (mysql-connector-java-5.1.25-bin.jar).

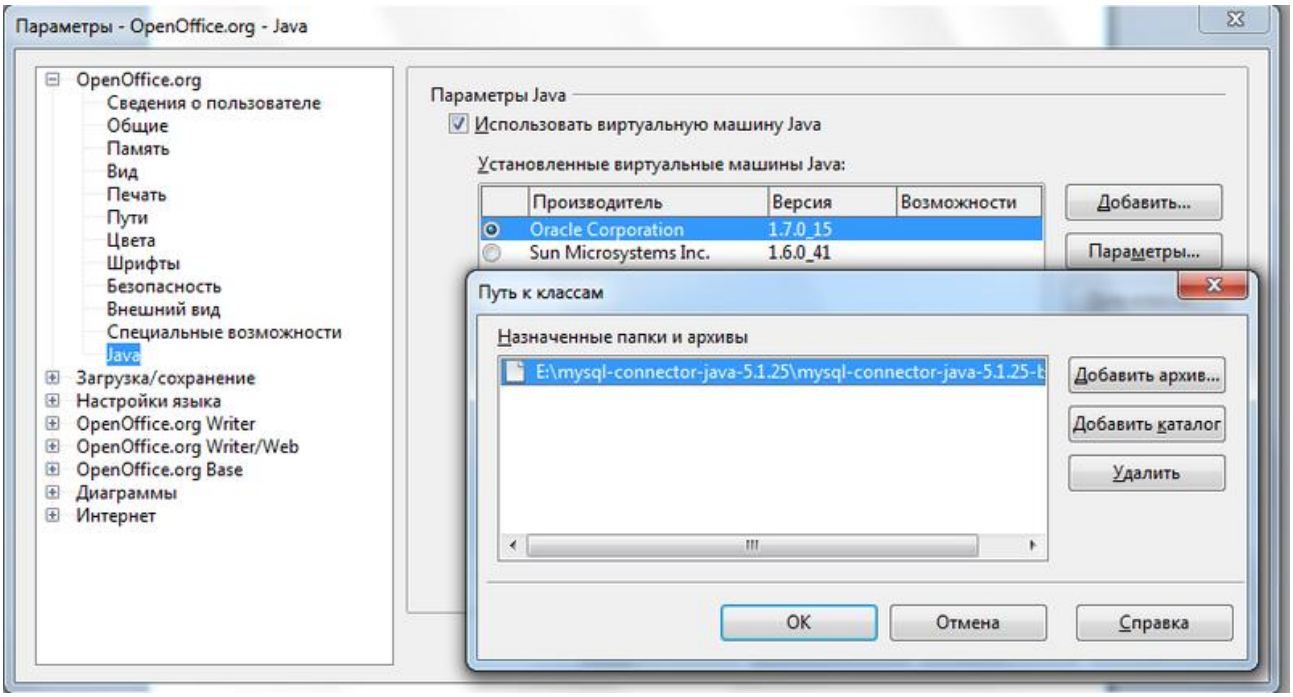

Рис. 4.4. Встановлення драйвера JDBC для MySQL

Далі перевіряємо роботу драйвера. Створюємо нову базу даних. Запускається майстер створення БД. Тут в якості джерела даних вибираємо JDBC. На другому кроці майстра необхідно виконати настройку з'єднання. Задаємо параметри налаштування з'єднання з базою MySQL через JDBC:

- Вказуємо Url джерела даних у форматі mysql://ім'я_сервера: 3306/ Імя_бази_данних_в_MySQL;
- Вказуємо клас драйвера JDBC: Для драйвера JDBC-MySQL це com.mysql.jdbc.Driver

Натискаємо кнопку "Перевірити клас". При натисканні кнопки відбувається перевірка завантаження JDBC – драйвера (рисунок 4.5).

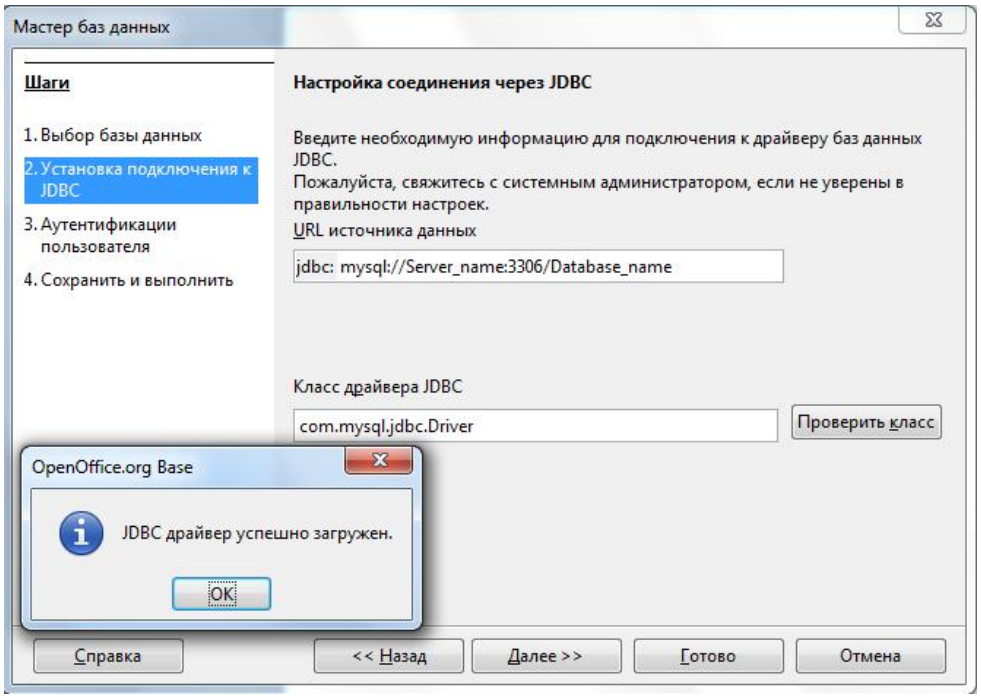

Рис. 4.5. Налаштування драйвера JDBC для MySQL

На наступному кроці пропонується ввести пароль та ім’я користувача з'єднання з MySQL для перевірки та встановлення з'єднання. Налаштування драйвера та проекту виконанено.

4.3. Інструкція користувача

Робота із системою управління запасами на підприємстві розпочинається із аутентифікації користувача. На рисунку 4.6 представлено форму аутентифікації користувача в системі.

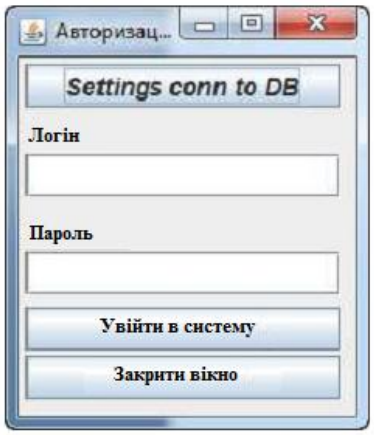

Рис. 4.6. Форма аутентифікації користувача

Адміністратор має права встановлювати настройки з'єднання з БД (рисунок 4.7).

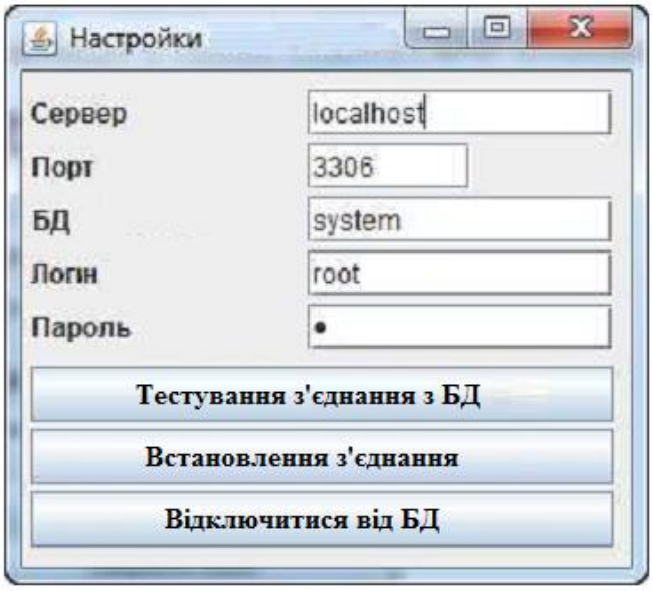

Рис. 4.7. Налаштування з’єднання з БД

Опис структури меню:

1. Управління замовленнями:
    - Виробничі замовлення;
    - Замовлення на поставку;
2. Складський облік:
    - Оприбуткування
    - Списання
    - Рух між складами;
3. Статистика:
    - Затримка поставок;
    - Статистика засобів.
4. Операції
    - Розрахунок запасних частин;
    - Розрахунок потреб на основі розкладу;
    - Розрахунок числа поставок;
    - Формування замовлень на поставку;

5. Налаштування

Каталоги, технологічні каталоги:

- Структура виробів;
- Каталог виробів;
- Каталог деталей;
- Каталог складальних одиниць;
- Каталог заготовок;
- Каталог комплектуючих виробів;
- Каталог типів елементів;
- Каталог матеріалів;
- Каталог груп матеріалів;

Управління прайс-листами, каталог складів. Каталог інструментів, пристосувань. Каталог обладнання. Управління безпекою. Імпорт даних. Головне меню автоматизованої системи управління запасами представлено на рисунку 4.8.

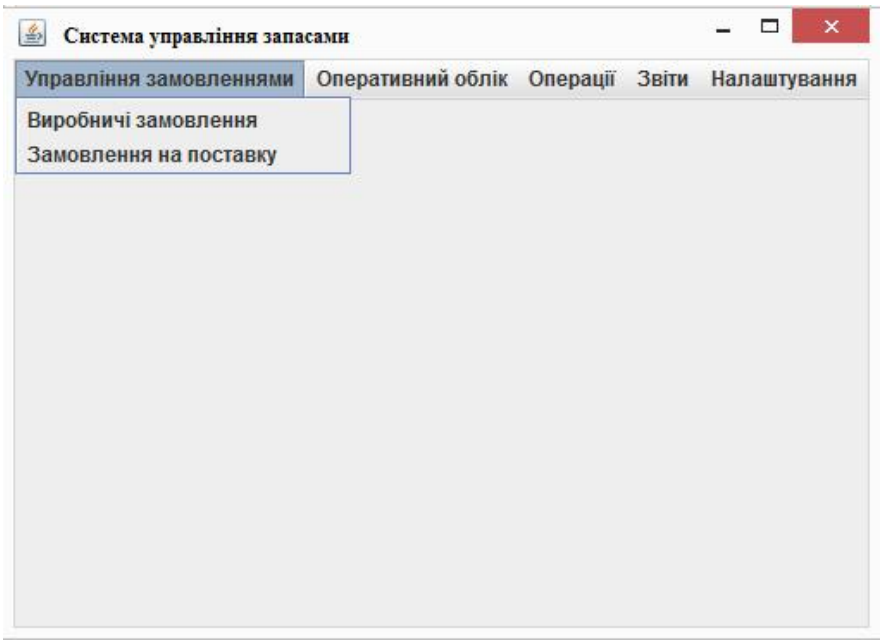

Рис. 4.8. Головне меню а системи

В системі розроблені функції ведення замовлень, каталогів, ведення складського обліку. Інтерфейси, в яких ведеться ця робота, представлені на рисунку 4.9-4.10.

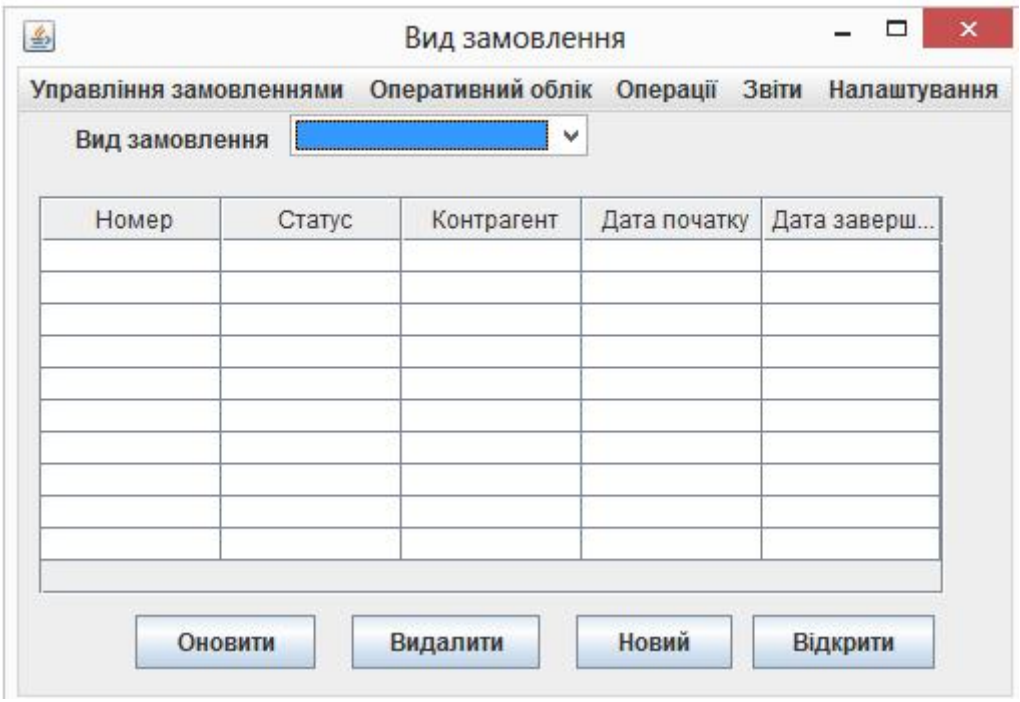

Рис. 4.9. Фомування замовлення на поставку матеріалів

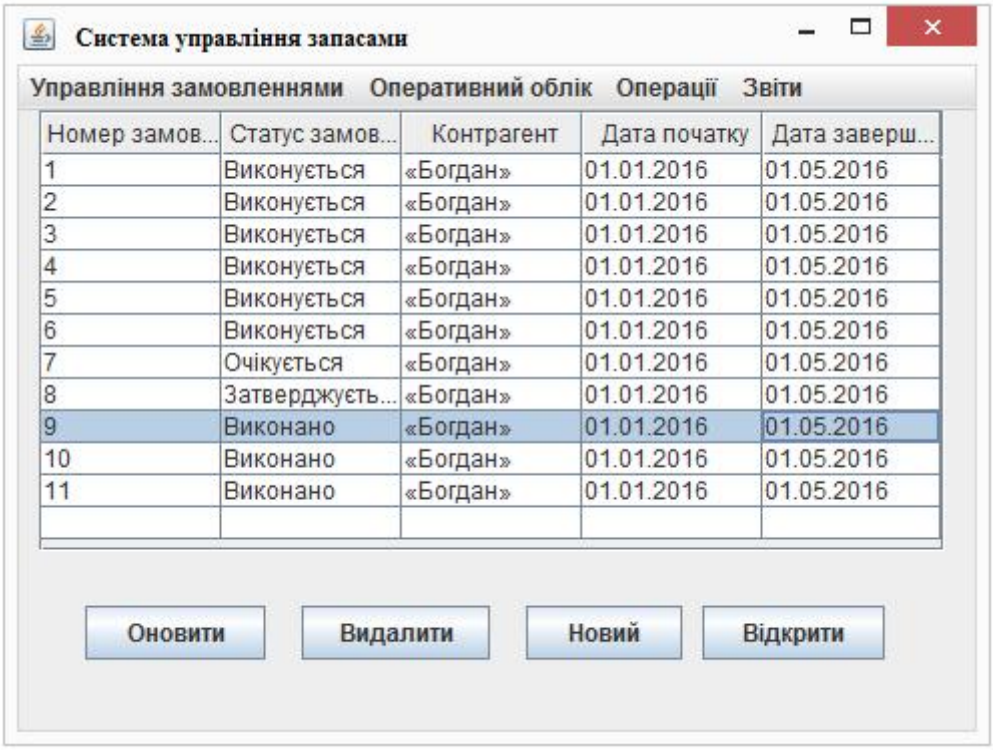

Рис. 4.10. Управління замовленнями

В системі чітко налагоджена процедура роботи з каталогами деталей, оскільки це впливає на якість формування замовлень. На рисунку 4.11 представлено інтерфейс з можливістю формування каталогу деталей, а також форму для оновлення приходу деталей (рисунок 4.12).

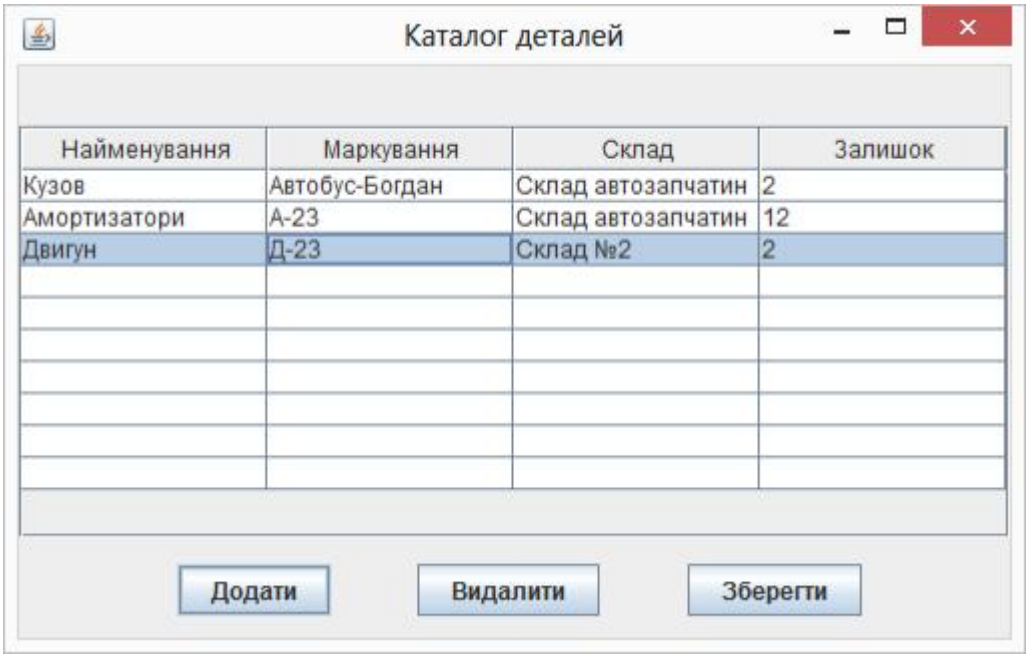

Рис. 4.11. Каталог деталей

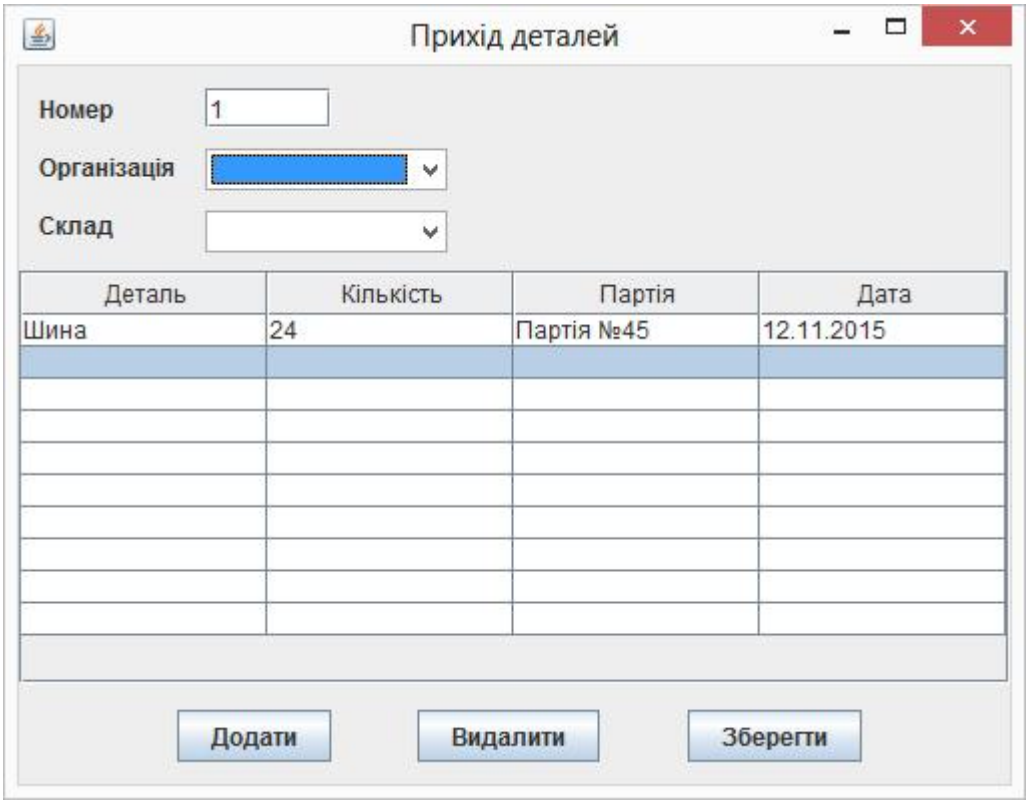

Рис. 4.12. Прихід деталей

Розрахунок потреб в матеріалах, комплектуючих за певним розкладом представлено на рисунку 4.13

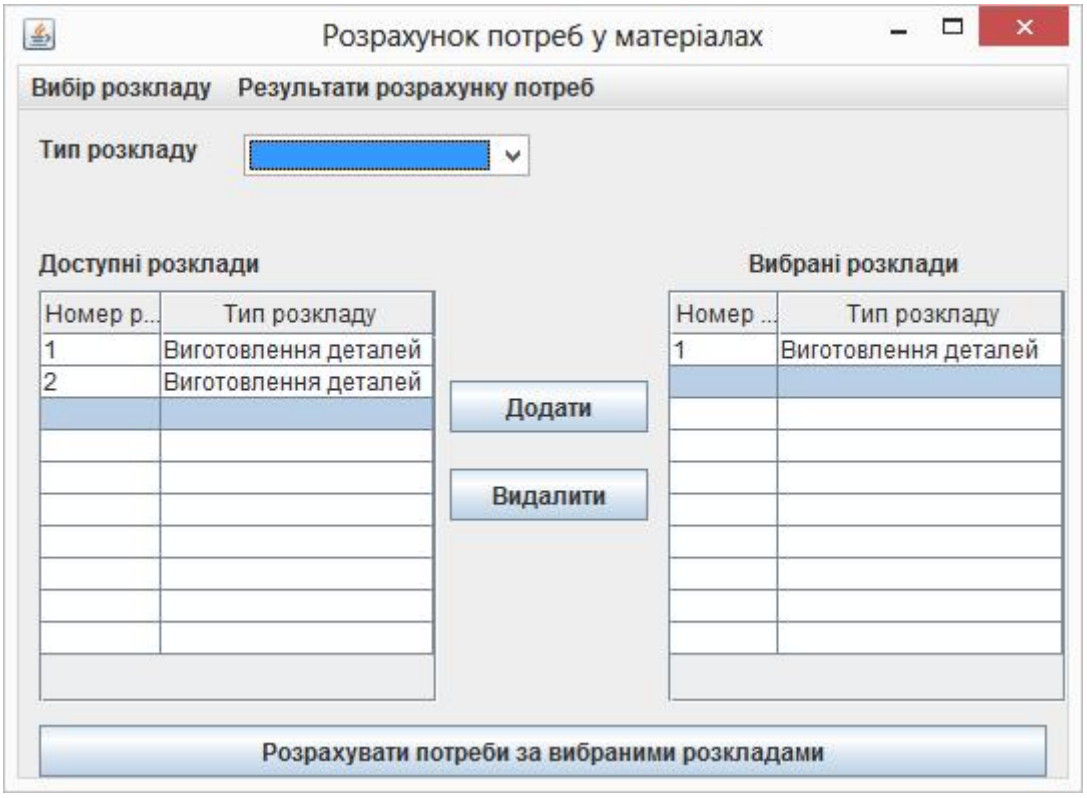

Рис. 4.13. Розрахунок потреб у матеріалах

Для імпорту даних з файлів розроблений функціонал який складається з наступних дій:

- вибір файлу розкладу (рисунок 4.14);
- процес перебігу імпорту файлів (рисунок 4.15);

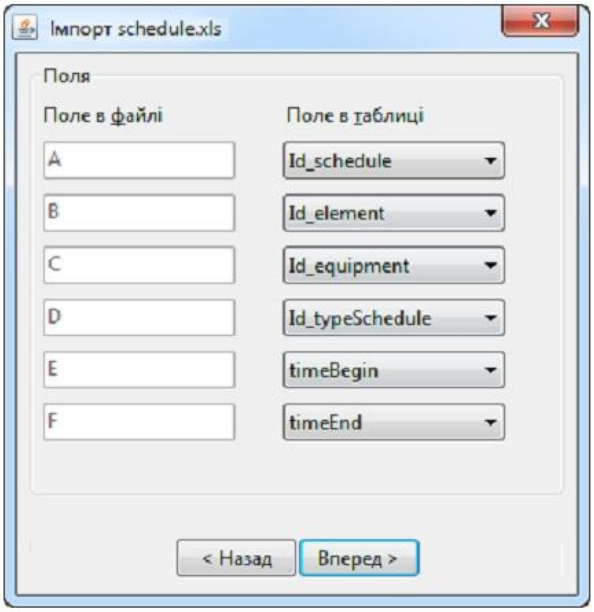

Рис. 4.14. Підсистема імпорту розкладів в систему

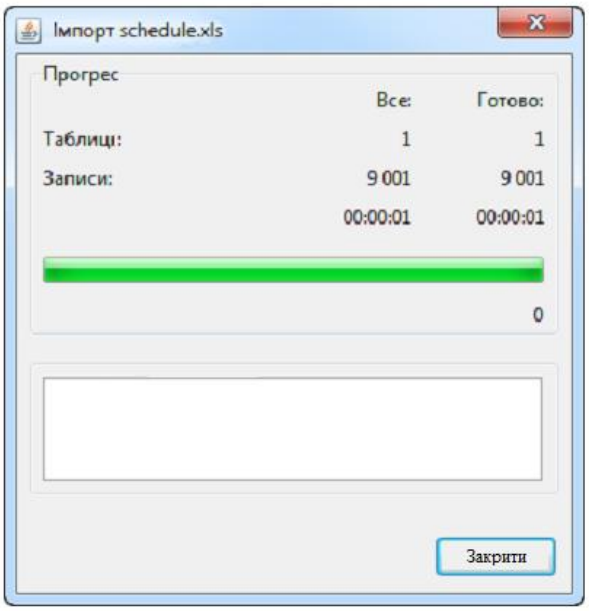

Рис. 4.15. Процес перебігу імпорту

Висновки до розділу 4

Здійснено опис процедур тестування та їхніх результатів, описані тест-вимоги до програмного забезпечення, а також виявлені дефекти. Розкрито питання встановлення та налаштування програмного забезпечення на сервері, а також вказані вимоги, дотримання яких необхідно для користування системою, описана інструкція користувача для роботи із системою.

ВИСНОВКИ

У відповідності з поставленими завданнями в даній роботі були вирішені наступні завдання:

• Проведено аналіз існуючих підходів до підвищення ефективності управління запасами машинобудівних виробництв;

• Проведено аналіз існуючих систем і моделей управління запасами;

• Визначено цільову функція сумарних витрат, пов'язаних зі зберіганням запасів матеріалів і комплектуючих, а також вартістю виконання замовлення;

• Досліджено цільову функція сумарних витрат і розроблений алгоритм пошуку найвідповіднішого числа поставок матеріалів та комплектуючих;

• Розроблено алгоритм розрахунку запасних деталей, заготовок, комплектуючих на основі моделі управління запасними частинами;

• Розроблено алгоритм формування замовлень на поставку матеріалів і комплектуючих на основі порівняння цін прайс-листів постачальників, введених в систему;

• Розроблено автоматизовану систему управління запасами з можливістю застосування на машинобудівних підприємствах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ландэ Д. В., Снарский А. А., Безсуднов И. В. Интернетика: Навигация в сложных сетях: модели и алгоритмы. — М.:Либроком (Editorial URSS), 2009. — 264 с. ISBN=978-5-397-00497-8.
2. Буров Є. В. Комп’ютерні мережі: підручник / Євген Вікторович Буров. — Львів: «Магнолія 2006», 2010. — 262 с. ISBN 966-8340-69-8
3. Ландэ Д. В., Снарский А. А., Безсуднов И. В. Интернетика: Навигация в сложных сетях: модели и алгоритмы. — М.:Либроком (Editorial URSS), 2009. — 264 с. ISBN=978-5-397-00497-8.
4. Комп’ютерні мережі: [навчальний посібник] / А. Г. Микитишин, М. М. Митник, П. Д. Стухляк, В. В. Пасічник. — Львів: «Магнолія 2006», 2013. — 256 с. ISBN 978-617-574-087-3
5. Инженерное программирование для проектирования программного обеспечения. -М.: Радио і связь, 1985, -512с.
6. Бойков.В., Савинков В.М. Проектирование баз данных информационных систем. М. Мир 1997
7. Бердтис А. Структуры данних. - М.: Статистика, 1974, - 408 с.
8. Горбань О.М., Бахрушин В.Є. Основі теорії систем та системного аналізу. - Запоріжжя, ГУ "ЗІДМУ", 2004, ISBN 966-8227-23-9
9. Майо Д. Самоучитель Microsoft Visual Studio 2010 = Microsoft Visual Studio 2010: A Beginner’s Guide (A Beginners Guide). — С.: «БХВ-Петербург», 2010. — С. 464. — ISBN 978-5-9775-0609-0
10. Алекс Макки Введение в .NET 4.0 и Visual Studio 2010 для профессионалов = Introducing .NET 4.0: with Visual Studio 2010. — М.: «Вильямс», 2010. — С. 416. — ISBN 978-5-8459-1639-6
11. Кен Хендерсон Професійне керівництво з SQL Server: структура та реалізація. — М.: Издательский дом «Вильямс», 2006. — С. 1056. ISBN 5-8459-0912-0

12. Чураков Михаил. Муравьиные алгоритмы [Електронний ресурс] / Михаил Чураков,Андрей Якушев. // Режим доступу: http://rain.ifmo.ru/cat/data/theory/unsorted/ant-algo-2006/article.pdf.
13. Бормашов Д. А. “Кластерный анализ текстов”: Дипломная работа [Електронний ресурс] /Д. А. Бормашов. Режим доступу: http://inf.tsu.ru/library/DiplomaWorks/CompScience/2006/bormashov/diplom.pdf.
14. Чубукова И.А. Data Mining БИНОМ. Лаборатория знаний, Интернет-университет информационных технологий - ИНТУИТ.ру, 2006.
15. Дюк В.А., Самойленко А.П. Data Mining: учебный курс. – СПб.: Питер, 2001.
16. Барсегян А. А., Куприянов М. С., Степаненко В. В., Холод И. И. Методы и модели анализа данных: OLAP и Data Mining. – СПб.: БХВ-Петербург, 2004. – 336 с.
17. Дивак М. П., Шпінталь М.Я., Козак О.Л., Струбицька І.П., Спільчук В.М., Піговський Ю.Р. Методичні рекомендації до виконання дипломної роботи освітньо кваліфікаційного рівня «бакалавр» студентам усіх форм навчання для напряму підготовки 6.050103 – «Програмна інженерія» // Тернопіль : ФОП Шпак П.П. - 2013. - 54 с.

ДОДАТОК А

ЛІСТИНГ ОСНОВНИХ МОДУЛІВ СИСТЕМИ

```
@Entity
@Table(name="CUSTOMER_ADDRESS")
@XmlRootElement(name="address")
@XmlAccessorType(XmlAccessType.FIELD)
public class Address {
   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private Long id;

   @XmlElement(required=true)
   protected int number;

   @XmlElement(required=true)
   protected String street;

   @XmlElement(required=true)
   protected String city;

   @XmlElement(required=true)
   protected String province;

   @XmlElement(required=true)
   protected String zip;

   @XmlElement(required=true)
   protected String country;

   public Address() { }

   // Getter and setter methods
   // ...
```

```
}
@Entity
@Table(name="CUSTOMER_CUSTOMER")
@NamedQuery(
   name="findAllCustomers",
   query="SELECT c FROM Customer c " +
        "ORDER BY c.id"
)
@XmlRootElement(name="customer")
@XmlAccessorType(XmlAccessType.FIELD)
public class Customer {
   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   @XmlAttribute(required=true)
   protected int id;

   @XmlElement(required=true)
   protected String firstname;

   @XmlElement(required=true)
   protected String lastname;

   @XmlElement(required=true)
   @OneToOne
   protected Address address;

   @XmlElement(required=true)
   protected String email;

   @XmlElement (required=true)
   protected String phone;

   public Customer() {...}

   // Getter and setter methods
```

```
    // ...
}
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="address" type="address"/>
  <xs:element name="customer" type="customer"/>

  <xs:complexType name="address">
    <xs:sequence>
      <xs:element name="id" type="xs:long" minOccurs="0"/>
      <xs:element name="number" type="xs:int"/>
      <xs:element name="street" type="xs:string"/>
      <xs:element name="city" type="xs:string"/>
      <xs:element name="province" type="xs:string"/>
      <xs:element name="zip" type="xs:string"/>
      <xs:element name="country" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="customer">
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
      <xs:element ref="address"/>
      <xs:element name="email" type="xs:string"/>
      <xs:element name="phone" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:int" use="required"/>
  </xs:complexType>
</xs:schema>
@Stateless
@Path("/Customer")
public class CustomerService {
    public static final Logger logger =
```

```
        Logger.getLogger(CustomerService.class.getCanonicalName());
@PersistenceContext
private EntityManager em;
private CriteriaBuilder cb;

@PostConstruct
private void init() {
   cb = em.getCriteriaBuilder();
}
...
@POST
@Consumes({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
public Response createCustomer(Customer customer) {

   try {
      long customerId = persist(customer);
      return Response.created(URI.create("/" + customerId)).build();
   } catch (Exception e) {
      logger.log(Level.SEVERE,
            "Error creating customer for customerId {0}. {1}",
            new Object[]{customer.getId(), e.getMessage()});
      throw new WebApplicationException(e,
            Response.Status.INTERNAL_SERVER_ERROR);
   }
}
...
private long persist(Customer customer) {
   try {
      Address address = customer.getAddress();
      em.persist(address);
      em.persist(customer);
   } catch (Exception ex) {
      logger.warning("Something went wrong when persisting the customer");
   }
   return customer.getId();
```

```
    }
@GET
    @Path("{id}")
    @Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
    public Customer getCustomer(@PathParam("id") String customerId) {
        Customer customer = null;

        try {
            customer = findById(customerId);
        } catch (Exception ex) {
            logger.log(Level.SEVERE,
                    "Error calling findCustomer() for customerId {0}. {1}",
                    new Object[]{customerId, ex.getMessage()});
        }
        return customer;
    }
    ...
    private Customer findById(String customerId) {
        Customer customer = null;
        try {
            customer = em.find(Customer.class, customerId);
            return customer;
        } catch (Exception ex) {
            logger.log(Level.WARNING,
                    "Couldn't find customer with ID of {0}", customerId);
        }
        return customer;
    }
@Named
@Stateless
public class CustomerBean {
    protected Client client;
    private static final Logger logger =
            Logger.getLogger(CustomerBean.class.getName());
```

```
@PostConstruct
private void init() {
   client = ClientBuilder.newClient();
}

@PreDestroy
private void clean() {
   client.close();
}

public String createCustomer(Customer customer) {
   if (customer == null) {
      logger.log(Level.WARNING, "customer is null.");
      return "customerError";
   }
   String navigation;
   Response response =
         client.target("http://localhost:8080/customer/webapi/Customer")
         .request(MediaType.APPLICATION_XML)
         .post(Entity.entity(customer, MediaType.APPLICATION_XML),
               Response.class);
   if (response.getStatus() == Status.CREATED.getStatusCode()) {
      navigation = "customerCreated";
   } else {
      logger.log(Level.WARNING, "couldn"t create customer with " +
            "id {0}. Status returned was {1}",
            new Object[]{customer.getId(), response.getStatus()});
      navigation = "customerError";
   }
   return navigation;
}

public String retrieveCustomer(String id) {
   String navigation;
   Customer customer =
```

```
            client.target("http://localhost:8080/customer/webapi/Customer")
            .path(id)
            .request(MediaType.APPLICATION_XML)
            .get(Customer.class);
        if (customer == null) {
            navigation = "customerError";
        } else {
            navigation = "customerRetrieved";
        }
        return navigation;
    }

    public List<Customer> retrieveAllCustomers() {
        List<Customer> customers =
            client.target("http://localhost:8080/customer/webapi/Customer")
            .path("all")
            .request(MediaType.APPLICATION_XML)
            .get(new GenericType<List<Customer>>() {});
        return customers;
    }
}
```