

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

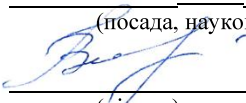
на здобуття ступеня вищої освіти "бакалавр"

на тему: Розробка АРМ продавця-консультанта салону побутової техніки

Виконав: студент групи

 Диченко Н.М.
(підпис)

Керівник роботи:

PhD
(посада, науковий ступінь вчене звання)
 Ворона М.В.
(підпис)

м. Миколаїв – 2026 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра інформаційних управляючих систем та технологій

Спеціальність 122 "Комп'ютерні науки"

Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

к.т.н, доц. Гайда А.Ю.

" ____ " ____ 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ на здобуття ступеня вищої освіти "бакалавр"

Студенту Диченко Наталії Миколаївні

(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка АРМ продавця-консультанта салону побутової техніки

Керівник роботи PhD Ворона Михайло Владиславович

Затверджені наказом ректора № 270-уч від "31" березня 2026 року.

2. Термін подання роботи: 22 червня 2026 р.

3. Вихідні дані до проєкту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем,

4. Перелік питань, що належать до розробки (найменування розділів) Вступ. 1 Аналіз предметної області і постановка задачі. 1.1 Загальний опис проблеми. 1.2 Аналіз підприємства для якого виконується автоматизація. 1.3 Аналіз існуючих рішень. 2 Розробка проєктних рішень. 2.1 Інформаційне забезпечення. 2.2 Програмне забезпечення. 3 Реалізація проєкту автоматизованої системи. 4 Охорона праці. Додатки

5. Перелік презентаційних матеріалів Структура АРМ менеджера інформаційної системи магазину мобільного зв'язку. Структура інформаційної системи та АРМ менеджера магазину мобільного зв'язку, що пропонується. Діаграма прецедентів АРМ менеджера. Концептуальна модель даних ІС. Логічна модель даних. Діаграма класів системи. Інтерфейс користувача. Діаграма розгортання системи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Ворона М.В.	02.02.2026	23.02.2026
2	Ворона М.В.	23.02.2026	20.03.2026
3	Ворона М.В.	20.03.2026	24.04.2026
4		24.04.2026	01.06.2026

7. Дата видачі завдання 02 лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

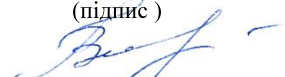
№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	02.02.2026	
2	Постановка задачі	16.02.2026	
3	Розробка концепції	23.02.2026	
4	Розробка проекту ІС	23.03.2026	
5	Реалізація проєкту	27.04.2026	
6	Розробка питань з охорони праці	01.06.2026	
7	Розробка графічних матеріалів	05.06.2026	
8	Подання роботи на перевірку на плагіат	08.06.2026	
9	Подання роботи рецензенту	15.06.2026	
10	Подання роботи на захист	22.06.2026	

Студент


(підпис)

Диченко Н.М.
(ПІБ)

Керівник роботи


(підпис)

Ворона М.В.
(ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена вдосконаленню управлінських процесів та розробленню автоматизованого робочого місця продавця-консультанта салону побутової техніки. Метою дослідження є підвищення ефективності операційної діяльності підприємства шляхом автоматизації основних функцій обліку, консультування клієнтів і підтримки процесу продажу, що сприятиме оптимізації роботи персоналу та зростанню економічних показників організації.

У роботі визначено загальну структуру управління салоном побутової техніки, сформульовано функціональні та нефункціональні вимоги до інформаційної системи, розглянуто можливі концепції її побудови та обґрунтовано вибір найбільш доцільного варіанта. Розроблено технічне завдання, здійснено проектування ключових рішень щодо інформаційного, програмного й технічного забезпечення, а також сформовано робочу документацію, необхідну для реалізації та подальшого супроводу програмного продукту.

Кваліфікаційну роботу викладено на 108 аркушах машинописного тексту. Робота містить 11 рисунків, 4 додатки, а також список використаних джерел у кількості 24 найменувань.

Ключові слова: АРМ, магазин, побутова техніка, Java, SQL.

ABSTRACT

The qualification thesis is devoted to the improvement of management processes and the development of an automated workplace for a sales consultant in a household appliances store. The purpose of the study is to increase the efficiency of the enterprise's operational activities by automating the main functions of accounting, customer consulting, and sales process support, thereby contributing to the optimization of staff performance and the improvement of the organization's economic indicators.

The thesis defines the general management structure of a household appliances store, formulates the functional and non-functional requirements for the information system, examines possible concepts for its development, and substantiates the selection of the most appropriate solution. The technical specification has been developed, key design decisions regarding information, software, and technical support have been made, and working documentation required for the implementation and further maintenance of the software product has been prepared.

The qualification thesis is presented on 108 pages of typed text. It includes 11 figures, 4 appendices, and a list of references comprising 24 sources.

Keywords: workstation, shop, household appliances, Java, SQL.

ЗМІСТ

ВСТУП.....	
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ.....	
1.1 Загальний опис проблеми.....	
1.2 Аналіз підприємства для якого виконується автоматизація.....	
1.3 Аналіз існуючих рішень.....	
Висновки до розділу 1.....	
2 РОЗРОБКА ПРОЄКТУ АВТОМАТИЗОВАНОЇ СИСТЕМИ.....	
2.1 Інформаційне забезпечення.....	
2.2 Програмне забезпечення.....	
Висновки до розділу 2.....	
3 РЕАЛІЗАЦІЯ ПРОЄКТУ АВТОМАТИЗОВАНОЇ СИСТЕМИ.....	
4 ОХОРОНА ПРАЦІ.....	
ДОДАТКИ.....	

ВСТУП

Дипломна робота присвячена розробці АРМ продавця-консультанта у структурі існуючої системи на прикладі магазину мобільного зв'язку, запропоновані різні методи для поліпшення ефективності роботи магазину, розроблена покращена система автоматизації робочого місця продавця для більш стабільного і постійно зростаючого прибутку магазину.

Останні роки ознаменовані стрімким зростанням попиту на мобільний зв'язок. У зв'язку з цим з'явилася величезна кількість точок, що здійснюють торгівлю мобільними пристроями та аксесуарами. Тим не менш, відповідно до думки фахівців, цей ринок до цих пір залишається повністю не насиченим. Особливо, якщо мова йде про регіони. Прикладом такого салону і потужної мережі магазинів може послужити магазин мобільного зв'язку та побутової техніки «Валдіс».

В даний час на фірмі продажів товару і пошуку клієнтів засновано в основному на паперовій підготовці пропозицій та їх розсилка, основними недоліками якого є:

- великі часові витрати на передачу інформації від одного місця в інше;
- помилки, які виникають через відсутність жорсткого контролю, який найбільш повно може бути здійснений тільки за допомогою ЕОМ;
- труднощі під час пошуку та ліквідації помилок, допущених на ранніх стадіях підготовки реклами.

У зв'язку з цим, застосування сучасний засобів проєктування систем в даному випадку є найбільш доцільним, так як ці кошти націлені на усунення саме цих недоліків, а також мають ряд інших додаткових переваг, суть яких буде розкрито далі.

Метою даного дипломного проєкту є розробка Web-сайту з функціями Internet-магазину для магазину мобільного зв'язку.

Дана розробка дозволити спростити та суттєво прискорити процес представлення фірми на ринку аналогічних послуг.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Загальний опис проблеми

Згідно з результатами маркетингових досліджень, значна частка рішень щодо придбання товарів приймається безпосередньо в місці продажу. Перебуваючи в торговельному залі, споживач зазнає впливу товарної викладки, рекламних матеріалів, музичного супроводу та інших чинників, що можуть стимулювати здійснення незапланованих покупок. Водночас у сучасних умовах дедалі більшого поширення набуває реалізація товарів через інтернет-магазини, які забезпечують зручний спосіб вибору, замовлення та придбання продукції без необхідності відвідування фізичної торговельної точки. У цьому розділі розглянуто та порівняно два зазначені підходи до продажу товарів, а також проаналізовано ефективність відповідних систем реалізації продукції.

У разі здійснення продажу безпосередньо у магазині сучасні підприємства роздрібно́ї торгівлі застосовують комплекс інструментів, спрямованих на збільшення тривалості перебування споживача в торговельному залі та активізацію його купівельної поведінки.

Для забезпечення ефективного продажу товарів у традиційному магазині необхідно реалізувати низку організаційних і маркетингових заходів, спрямованих на залучення покупців та формування сприятливих умов для прийняття рішення про купівлю:

1. **Забезпечення чистоти.** Одним із першочергових чинників, на які звертає увагу споживач, є санітарний стан торговельного залу, зокрема чистота підлоги, відсутність пилу на полицях і товарах, а також охайний зовнішній вигляд продавців і консультантів. Доцільно забезпечити належний рівень чистоти ще до розміщення торговельного обладнання та викладки товарів,

оскільки недотримання цих вимог може негативно вплинути на сприйняття магазину споживачем.

2. Забезпечення огляду товарного асортименту. Споживач, заходячи до торговельного залу, повинен мати можливість швидко зорієнтуватися у розміщенні товарних груп. Для цього використовуються покажчики товарних секцій, логічне зонування простору та візуально доступна викладка продукції.

3. Забезпечення доступності товару. Товар має бути розміщений таким чином, щоб покупець міг безперешкодно його оглянути та взяти без докладання значних зусиль. При цьому необхідно враховувати призначення товару та характеристики цільової аудиторії. Наприклад, товари для дітей доцільно розташовувати на рівні, зручному для їхнього огляду дитиною.

4. Забезпечення комфортності перебування у магазині. Комфорт є комплексним поняттям, що залежить від індивідуальних характеристик споживача, зокрема його психографічних особливостей. Водночас можна виокремити низку загальних вимог, дотримання яких сприяє формуванню позитивного споживчого досвіду:

1. Зручність переміщення у торговельному залі. Покупці повинні мати можливість вільно пересуватися між товарними секціями, не створюючи перешкод іншим відвідувачам і не зачіпаючи торговельне обладнання.

2. Музичний супровід у торговельному залі. Музика може виступати дієвим інструментом впливу на поведінку споживачів. Динамічний музичний супровід сприяє активнішому переміщенню покупців і прискоренню процесу прийняття рішень, тоді як повільна, зокрема класична, музика створює умови для тривалішого перебування у магазині. Вибір музичного оформлення має відповідати позиціонуванню торговельного закладу та особливостям цільової аудиторії.

3. Якісне обслуговування. Персонал магазину повинен бути готовим своєчасно надати покупцеві необхідну консультаційну допомогу. Ввічливість, компетентність і здатність працівників надати повну інформацію про товар позитивно впливають на рівень довіри споживачів та формування їхньої лояльності до торговельного закладу.

4. Надання додаткових послуг. Покупці можуть відвідувати магазин не лише з метою придбання товарів, а й для отримання консультацій, інформаційної підтримки або супутніх сервісів. Тому наявність додаткових послуг сприяє підвищенню рівня задоволеності споживачів і розширенню функціональних можливостей торговельного закладу.

5. Наявність допоміжного обладнання. До елементів, що полегшують процес купівлі, належать купівельні кошики, каталоги, місця для відпочинку, підставки для сумок, а також спеціально обладнані зони для дітей. Такі засоби підвищують зручність перебування покупців у магазині.

7. Забезпечення наповненості полиць. Важливим чинником є формування у споживача враження постійної наявності необхідного асортименту. З цією метою полиці мають регулярно поповнюватися товарами, що відповідають потребам покупців.

8. Забезпечення інформативності товарів. Упаковка товару повинна містити повну та зрозумілу інформацію, необхідну споживачеві для прийняття обґрунтованого рішення про купівлю.

9. Забезпечення зрозумілості ціни. Цінова інформація має бути доступною, однозначною та зрозумілою для споживача.

Реалізація зазначеного комплексу заходів потребує додаткового часу на передпродажну підготовку та може зумовлювати підвищення вартості товарів. У зв'язку з цим доцільно розглянути альтернативний спосіб реалізації продукції — продаж мобільних телефонів через інтернет-магазини.

Діяльність менеджерів інтернет-магазинів ускладнюється значною кількістю товарних позицій і замовлень, необхідністю оперативного опрацювання заявок, перевірки наявності товарів, оновлення асортименту, вилучення неактуальних позицій та редагування описів. Такі процеси потребують значних часових і трудових ресурсів. Тому інформаційна система, що розробляється, призначена для оптимізації роботи магазину з продажу мобільних телефонів. До основних завдань адміністратора належать надання клієнтам інформації про наявні товари, приймання замовлень та здійснення розрахунків із постачальниками телефонів.

Актуальність розв'язуваної задачі зумовлена широким застосуванням засобів автоматизації у сфері торгівлі — від баз даних до сучасних технологій ідентифікації товарів. Попри те, що в Україні такі підходи ще не набули повсюдного поширення, спостерігається поступовий розвиток програмних продуктів, спрямованих на автоматизацію торговельних процесів. Використання Інтернету в комерційній діяльності є перспективним напрямом, оскільки дає змогу скоротити витрати, підвищити рівень доходів і забезпечити покупцям зручніші умови придбання товарів. У межах розроблюваної системи необхідна інформація зберігатиметься у взаємопов'язаних таблицях реляційної бази даних. Клієнт під час оформлення замовлення матиме змогу обирати службу доставки з урахуванням термінів і вартості, а також форму оплати. У системі управління базами даних передбачається ведення обліку постачальників, обсягів поставлених товарів у кількісному та грошовому вираженні, а також обсягів виконаних доставок різними службами. Крім того, для кожного клієнта може розраховуватися знижка залежно від загальної суми здійснених покупок, а також відкриватися персональний грошовий рахунок для оплати та поповнення.

Запропонований проєкт сприятиме спрощенню внесення та оновлення персональних даних, а також зменшенню ризику виникнення суперечливої інформації й інформаційних аномалій. Він забезпечить зручність і зрозумілість процесу купівлі для клієнтів, а також уніфікує й спростить обслуговування для адміністраторів. У контексті розвитку інформаційних технологій особливого значення набуває здатність підприємства швидко адаптуватися до змін динамічного конкурентного середовища. Ефективне використання інформаційних ресурсів дає змогу підвищити якість управлінських рішень, оптимізувати організаційну структуру та забезпечити належний рівень взаємодії між підрозділами підприємства. Частина комплексів завдань, пов'язаних з управлінням персоналом і організаційними процесами, може ефективно підтримуватися сучасними інформаційними технологіями. Отже, результативність функціонування системи управління підприємством значною мірою залежить від якості, повноти та актуальності інформаційних баз.

Інформаційні бази підприємства становлять сукупність даних про стан керованої системи та тенденції її розвитку, відображених через кількісні й якісні характеристики відповідних процесів. Будь-яке управлінське рішення має ґрунтуватися на об'єктивному розумінні поточної ситуації, чинників впливу та взаємозв'язків між ними. Інформація забезпечує реалізацію основних функцій управління, зокрема планування, організації, контролю, регулювання та розподілу ресурсів.

Багато процесів, що визначають кадрову та організаційну ситуацію на підприємстві, мають динамічний характер: змінюється чисельність персоналу, його структура за віком, стажем роботи, кваліфікацією та іншими ознаками; відбувається внутрішній рух працівників між структурними підрозділами; оновлюються професійні ролі та рівень компетентності. У цьому контексті інформація виступає засобом контролю відповідності фактичної структури

трудових ресурсів потребам підприємства, а також своєчасного укомплектування вакантних посад. Отже, основними вимогами до кадрової інформації є оперативність, достовірність і надійність.

1.2 Аналіз підприємства для якого виконується автоматизація

Щоб точніше сформулювати аналіз предметної області, потрібно розпочати з досліджуваних об'єктів. Самим найголовнішим виступає діюче підприємство, в статуті якого передбачається наявність певних посад, котрі повинні зайняти кваліфіковані кадри. Підприємство без кадрів, це не підприємство, а пуста множина, тобто відсутність трудового потенціалу свідчить про непридатність підприємства і неможливість його функціонування взагалі. Наявність хоч однієї відповідальної особи в межах функціонування даного об'єкту, розкриває його функціональне єство: можливість існувати в прогресивному середовищі. Посади, що передбачені внутрішньою документацією підприємства можуть посідати кваліфіковані кадри, а можуть і ні, тому що, в трудовій діяльності трапляються непоодинокі випадки, коли працівники звільняються з тих чи інших причин, і займана цим працівником посада стає вакантною для іншої особи. Будь-який персонал характеризується таким показниками як стаж і освіта, що характеризують якісний потенціал кожного працівника. Зрозуміло і те, що кожне підприємство складається із структурних підрозділів, перед котрими стоїть виконання окремих видів діяльності. З цього, випливає, що кожен відділ повинен звітувати перед вищестоящою посадовою особою. Крім того, на підприємстві можуть виникати такі явища, як відсутність працівників, з тих чи інших причин: лікарняне, відпустка, відрядження, перекваліфікація та ін.

Інформаційно-пошукова система може практично використовуватися в будь-якому діючому підприємстві, головним директором, замісником директора, завідувачами та керівниками, замісниками керівників структурних підрозділів, керівниками цехів, бригад – тими котрі постійно звітують перед вищестоящою адміністративною особою; колегіальними органами, менеджерами, котрі безпосередньо пов'язані з прийняттям управлінських рішень. Метою використання даної системи є спрощення та скорочення часових затрат на пошук відповідної інформації в існуючій базі даних.

У магазині задіяні наступні працівники: відповідальний за магазин - директор, у нього в підпорядкуванні відділ закупівель, відділ продажу, адміністратор сайта та бухгалтер. Кожен з них виконує свої специфічні функції. Відділ закупівель займається пошуком відповідного товару та його асортименту, відділ продажу займається реалізацією товару та пошуком нових клієнтів, адміністратор сайта займається рекламою в мережі Інтернет, бухгалтер веде бухгалтерський облік магазину, Директор слідкує за тим щоб діяльність фірми була налагоджена найкращим чином.

Організаційна структура мережі магазинів побутової техніки представлена нижче.

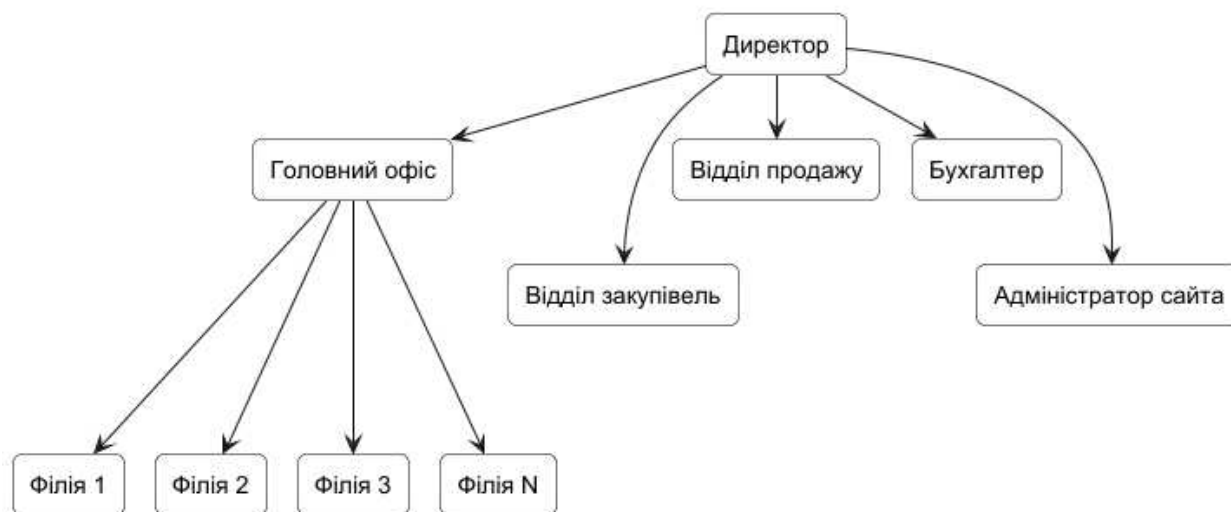


Рисунок 1.1 – Організаційна структура мережі магазинів побутової техніки

На підприємстві відбуваються наступні процеси:

Попередній аналіз предметної області свідчить про необхідність реалізації таких основних запитів до системи:

1. Каталог всіх товарів , які пропонує магазин;
2. Виявлення всіх товарів, котрі існують на складі;
3. Отримання списку товарів по коду телефону;
4. Виведення ціни товару;
5. Замовлення вибраного товару;
6. Оформлення заказу.

1.2.1 Структура існуючої інформаційної системи

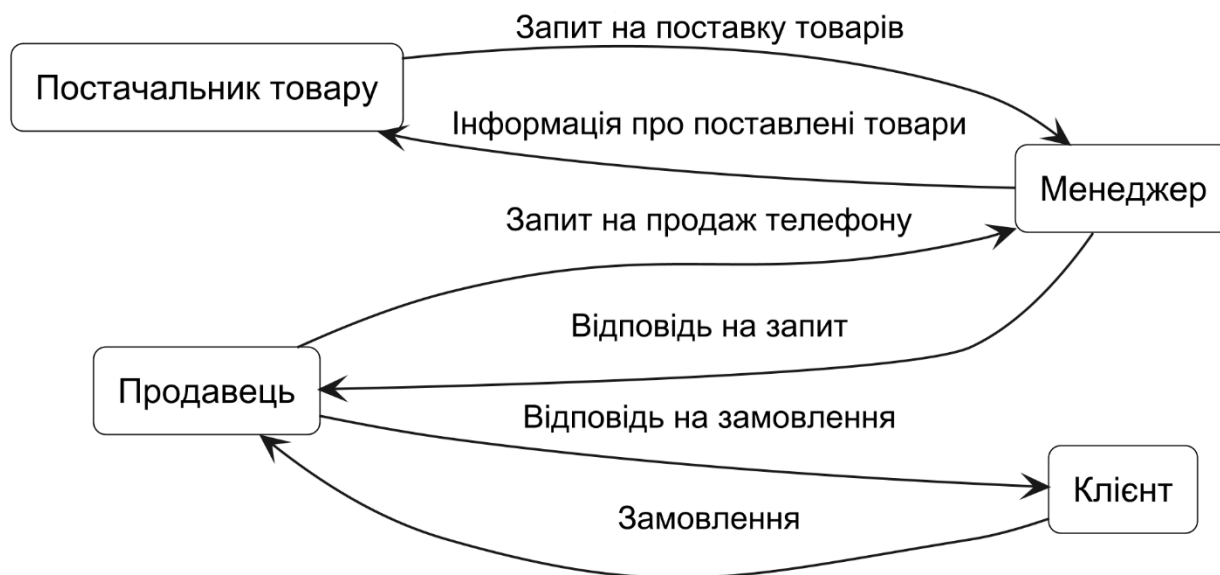


Рисунок 1.2- Структура існуючої інформаційної системи

На рисунку 1.2 показана структура існуючої системи. Як бачимо з Рисунку 1.1, ця система своєю структурою і функціональністю не може відповідати потребам роботи сучасної фірми.

Структура інформаційної системи, що пропонується, представлена на Рисунку 1.3

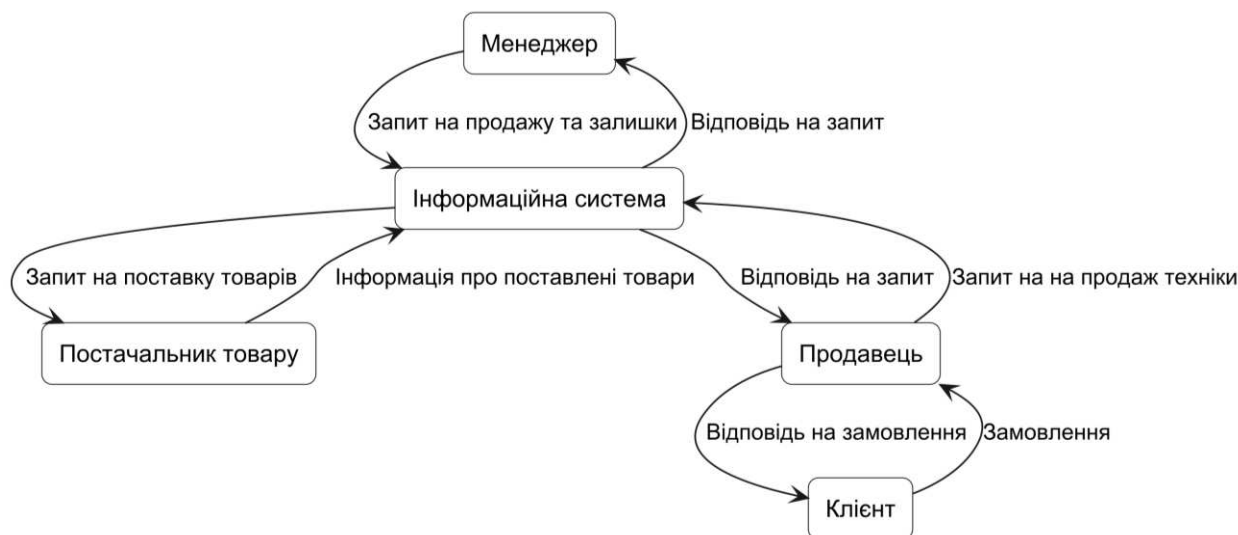


Рисунок 1.3 – Схема запропонованої системи

1) Менеджер-керуючий магазином. Обов'язками менеджера є планування, розробка і впровадження стратегічних планів для магазину, організація економічно обґрунтованої операційної діяльності, також в його обов'язки входить слідкування за правильною роботою системи, розподіл права доступу до неї. Права менеджера отримувати необхідну інформацію через інформаційну систему про кількість проданого і отриманого товару.

2) Постачальник товару – це юридична або фізична особа, що поставляє товари замовникам. Постачальник здійснює підприємницьку діяльність відповідно до умов укладеного договору поставки в обумовлений строк. У даній схемі він використовуючи інформаційну систему отримує замовлення від продавця, після перевірки менеджера магазину, і виконує поставку товару згідно чинного договору.

3) Інформаційна система – є сукупність технічного, програмного, організаційного забезпечення, та призначена для того, щоб своєчасно забезпечувати належних людей належною інформацією. В даній схемі їй

користуються всі учасники торгової мережі, отримуючи, аналізуючи і повертаючи необхідну інформацію.

4) Працівник – це особа яка відповідає на запит клієнта, проводить повну консультацію з даного товару і оформляє безпосередньо саме замовлення.

5) Клієнт – особа яка робить замовлення на товар, з певними вимогами до нього.

1.3 Аналіз існуючих рішень

На підставі проведеного дослідження об'єкта автоматизації визначено основні функціональні напрями, які мають бути реалізовані в межах розроблюваної інформаційної системи. До них належать представлення підприємства у глобальній мережі Інтернет, розміщення відомостей про товари, послуги, вартість робіт і результати діяльності, а також забезпечення можливості формування замовлень за допомогою спеціалізованої електронної форми. Крім того, на основі отриманих замовлень система має забезпечувати підготовку відповідних відомостей щодо необхідних матеріалів, товарних позицій або супутніх ресурсів.

Для вибору раціонального способу реалізації поставленого завдання доцільно розглянути кілька концептуальних підходів до створення та впровадження веборієнтованої інформаційної системи. У практиці автоматизації діяльності підприємств найчастіше застосовують такі основні варіанти:

- придбання готового програмного комплексу для управління вебсайтом та його подальше впровадження на підприємстві;
- залучення зовнішніх фахівців або спеціалізованої організації у сфері розроблення вебсайтів та інформаційних систем;

- проєктування вебсайту, що відповідає визначеним вимогам, із залученням штатних фахівців підприємства або працівників підрозділу автоматизованих систем управління.

Подальший аналіз зазначених підходів дає змогу оцінити їхні переваги, недоліки, економічну доцільність, рівень адаптованості до потреб підприємства та складність подальшого супроводу. Саме порівняння цих чинників є підґрунтям для вибору найбільш ефективного рішення.

На сучасному етапі розвитку інформаційних технологій існує значна кількість систем управління вебсайтами, які відрізняються функціональними можливостями, архітектурою, вартістю впровадження, рівнем безпеки та вимогами до технічного обслуговування. Такі системи представлені як зарубіжними, так і вітчизняними розробниками, що дає змогу обирати рішення відповідно до масштабу підприємства, специфіки його діяльності та наявних ресурсів.

1.3.1 Готові системи управління салоном побутової техніки

Управління салоном побутової техніки належить до складних організаційно-інформаційних процесів, оскільки поєднує операції роздрібної торгівлі, складського обліку, взаємодії з постачальниками, консультування клієнтів, оформлення замовлень, гарантійного супроводу та аналізу результатів продажу. Для підприємств такого типу особливо важливими є точність інформації про товарні залишки, актуальність цін, можливість швидкого пошуку продукції за технічними характеристиками, підтримка різних каналів продажу та формування звітності для прийняття управлінських рішень. У зв'язку з цим доцільно розглянути наявні програмні системи, які можуть бути використані для автоматизації діяльності салону побутової техніки, а також оцінити їхню відповідність функціональним потребам підприємства.

Аналіз готових рішень доцільно проводити за кількома основними критеріями. По-перше, система має забезпечувати ведення електронного каталогу товарів із можливістю фіксації детальних характеристик побутової техніки, зокрема виробника, моделі, класу енергоефективності, габаритів, гарантійного строку, комплектації та інших параметрів. По-друге, важливим є підтримання оперативного складського обліку, оскільки наявність або відсутність товару безпосередньо впливає на якість обслуговування клієнта. По-третє, програмне забезпечення повинно підтримувати продажі, повернення, резервування товарів, роботу з постачальниками, формування замовлень на поповнення складу та підготовку аналітичних звітів. Окреме значення має можливість інтеграції з вебсайтом, платіжними сервісами, системами бухгалтерського обліку та торговельним обладнанням.

У сучасній практиці автоматизації торговельних підприємств можна виокремити кілька класів програмних продуктів. До першої групи належать ERP-системи, що забезпечують комплексне управління підприємством і охоплюють фінансовий, складський, закупівельний, збутовий та аналітичний контури. До другої групи належать POS-системи, орієнтовані насамперед на обслуговування продажів у торговельній точці, фіксацію касових операцій, облік платежів і швидке формування чеків. Третю групу становлять системи управління інтернет-магазином і контентом, які забезпечують представлення товарів у мережі Інтернет, приймання замовлень і підтримку електронного каталогу. Крім того, у торговельній діяльності застосовуються CRM-системи, які зосереджені на взаємодії з клієнтами, історії звернень, програмах лояльності та персоналізованих пропозиціях.

Для українських підприємств одним із поширених напрямів автоматизації є використання програмних продуктів сімейства BAS. Зокрема, BAS ERP призначена для побудови комплексних інформаційних систем

управління багатoproфільними підприємствами та охоплює управління продажами, закупівлями, складом, запасами, фінансами, персоналом і звітністю. Для торговельних підприємств також може застосовуватися BAS Управління торгівлею, що орієнтована на планування та аналіз продажів, управління поставками, складськими запасами, взаєморозрахунками з покупцями й постачальниками, ціновою політикою та інтеграцію з торговельним обладнанням.

Перевагою BAS-рішень є їхня адаптація до українського середовища ведення бізнесу, підтримка регламентованого обліку, наявність типових конфігурацій і можливість доопрацювання під специфічні потреби підприємства. Для салону побутової техніки такі системи можуть забезпечити централізоване управління товарними залишками, оформлення закупівель, облік продажів, аналіз оборотності товарів і підготовку управлінської звітності. Водночас їхнє впровадження може потребувати значних фінансових витрат, залучення спеціалістів із налаштування, навчання персоналу та адаптації бізнес-процесів до логіки типової конфігурації. Крім того, для невеликого салону повний функціонал ERP-системи може бути надлишковим, що знижує економічну доцільність такого рішення на початковому етапі автоматизації.

Іншу групу становлять хмарні та локальні POS-рішення, серед яких у світовій практиці поширені Square for Retail, Lightspeed Retail, Shopify POS, Epos Now та подібні системи. Їх основним призначенням є підтримка операцій продажу, облік товарів у торговельній точці, робота з касою, сканерами штрихкодів, знижками, поверненнями, клієнтськими картками та базовою аналітикою. Такі системи зазвичай мають зручний інтерфейс, швидко впроваджуються та можуть бути корисними для підприємств, яким потрібна насамперед автоматизація фронт-офісних операцій. Для салону побутової

техніки POS-система може бути ефективною у частині реєстрації продажів, контролю залишків і формування первинних звітів про реалізацію товарів.

Разом із тим POS-рішення мають певні обмеження. Вони не завжди забезпечують повноцінне управління закупівлями, складною структурою характеристик товарів, сервісним обслуговуванням, гарантійними зверненнями та багаторівневою аналітикою. Якщо салон побутової техніки працює з широким асортиментом, кількома постачальниками, різними каналами продажу та потребує детального аналізу прибутковості, використання лише POS-системи може бути недостатнім. У такому разі POS доцільно розглядати як складову ширшої інформаційної архітектури, що інтегрується з ERP, CRM або системою управління інтернет-магазином.

Окрему увагу слід приділити системам електронної комерції, таким як Shopify, WooCommerce, OpenCart, Magento та інші платформи, що дають змогу створювати інтернет-магазини з електронним каталогом, кошиком, формою замовлення та інструментами адміністрування контенту. Для салону побутової техніки наявність веборієнтованого каналу продажу є важливою, оскільки клієнти часто порівнюють товари за характеристиками, ціною, наявністю, умовами доставки та гарантії ще до відвідування фізичного магазину. Перевагою таких платформ є можливість швидкого розміщення товарів, налаштування категорій, публікації описів, зображень, відгуків і проведення маркетингових акцій.

Недоліком платформ електронної комерції є те, що вони зазвичай не охоплюють повного циклу внутрішнього управління підприємством. Вони добре виконують функції представлення товарів і приймання замовлень, однак для повноцінного складського, фінансового й управлінського обліку потребують додаткових модулів або інтеграції з іншими системами. Для салону побутової техніки це означає необхідність синхронізації залишків, цін,

замовлень і даних про клієнтів між сайтом, складом, касою та бухгалтерським обліком. За відсутності такої інтеграції виникає ризик дублювання інформації, помилок у залишках, продажу відсутнього товару та несвоєчасного оновлення цін.

CRM-системи, зокрема Bitrix24, Zoho CRM, HubSpot CRM та інші подібні рішення, можуть використовуватися для підтримки взаємодії з клієнтами. У діяльності салону побутової техніки такі системи дають змогу зберігати історію звернень, фіксувати консультації, контролювати повторні контакти, формувати клієнтські сегменти, організовувати розсилки та підтримувати програми лояльності. Особливо важливими ці функції є для продажу технічно складних товарів, де клієнт часто потребує додаткової консультації, порівняння моделей, уточнення умов доставки, встановлення, гарантійного або післягарантійного обслуговування.

Однак CRM-система не є повноцінною заміною системи управління торговельними процесами. Її сильним боком є робота з клієнтськими даними, комунікаціями та продажними воронками, тоді як облік товарних залишків, закупівлі, складська логістика та касові операції зазвичай реалізуються частково або через інтеграції. Тому для салону побутової техніки CRM доцільно використовувати як допоміжний компонент, що підвищує якість клієнтського сервісу, але не вирішує всі завдання автоматизації підприємства.

Порівняння наведених класів програмних систем свідчить, що кожен із них орієнтований на певний рівень управління. ERP-системи забезпечують найбільш повне охоплення бізнес-процесів, але є складними й витратними у впровадженні. POS-системи ефективні для швидкого обслуговування покупців і фіксації продажів, проте мають обмежені можливості щодо стратегічного аналізу й комплексного управління підприємством. Платформи електронної комерції забезпечують представлення товарів у мережі Інтернет, однак

потребують інтеграції з внутрішніми обліковими системами. CRM-рішення посилюють клієнтоорієнтованість підприємства, але не замінюють складський і фінансовий облік.

Для салону побутової техніки важливою є не лише наявність окремих функцій, а й узгодженість інформаційних потоків між продажами, складом, постачанням, клієнтським сервісом і звітністю. Побутова техніка має значну кількість технічних характеристик, відмінності між моделями, різні гарантійні умови, залежність від постачальників і сезонні коливання попиту. Тому система управління повинна підтримувати структурований каталог, швидкий пошук, резервування товару, контроль залишків, аналіз продажів, облік замовлень і можливість подальшого розширення функціоналу.

Застосування готових програмних продуктів може бути доцільним за умови, що їхня функціональність відповідає реальним потребам підприємства, а витрати на придбання, налаштування й супровід є економічно обґрунтованими. Проте готові системи часто містять надлишкові модулі або, навпаки, не враховують специфіку конкретного салону, його асортиментну політику, порядок роботи з клієнтами, особливості формування замовлень і внутрішню організаційну структуру. У таких випадках підприємство змушене або адаптувати власні процеси до логіки програмного продукту, або замовляти додаткові доопрацювання, що збільшує вартість проєкту та строки впровадження.

Отже, аналіз існуючих систем засвідчує, що на ринку наявні різні програмні рішення для автоматизації торговельної діяльності: від комплексних ERP-платформ до спеціалізованих POS, CRM і веборієнтованих систем. Водночас для салону побутової техніки оптимальним є рішення, яке поєднує можливості товарного обліку, управління замовленнями, підтримки продажів, роботи з клієнтами та формування звітності без надмірної складності для

користувача. Саме тому розроблення власної або адаптованої інформаційної системи може бути обґрунтованим, якщо вона орієнтована на конкретні бізнес-процеси підприємства, забезпечує необхідний рівень функціональності та дає змогу зменшити витрати на використання зайвих модулів.

1.3.2 Система управління сайтом (контентом)

Система управління контентом вебсайту, або CMS, є програмним інструментом, що забезпечує адміністратору чи власнику вебресурсу можливість формувати його структуру, керувати інформаційним наповненням і підтримувати актуальність опублікованих матеріалів. За допомогою такої системи користувач може додавати нові сторінки, редагувати тексти, розміщувати зображення, створювати розділи, змінювати меню та оновлювати інформацію без безпосереднього втручання у програмний код. Це робить CMS доцільним рішенням для підприємств, які потребують оперативного, зручного та контрольованого управління власним вебпредставництвом.

Адміністративна панель CMS забезпечує централізований доступ до функцій редагування вебсайту. Вона дає змогу змінювати або додавати матеріали для різних мовних версій, керувати структурою сторінок, визначати права доступу користувачів і контролювати процес публікації. Після внесення та збереження змін оновлена інформація, як правило, відображається на сайті без значних часових затримок.

Системи управління вебсайтом виконують роль інструментів моделювання розгалуженої структури вебресурсу та організації його інформаційного наповнення. Їх використання не потребує від адміністратора глибоких знань програмування або HTML-верстки, оскільки більшість операцій здійснюється за допомогою візуального редактора, подібного до текстового процесора. Впровадження CMS також сприяє підвищенню

контрольованості змін, оскільки система дає змогу обмежувати доступ до окремих розділів і фіксувати дії користувачів.

Базова система управління сайтом зазвичай охоплює такі функціональні можливості:

- пошук інформації на вебсайті;
- збирання даних про клієнтів та потенційних споживачів;
- формування, редагування та опрацювання опитувань;
- аналіз відвідуваності вебсайту та поведінки користувачів.

До основних модулів системи управління вмістом сайту належать:

- модуль «Статті», призначений для додавання, редагування, планування та публікації сторінок сайту за допомогою зручного візуального інтерфейсу;

- модуль «Пункти меню», що забезпечує створення, редагування та впорядкування пунктів меню різних рівнів вкладеності;

- модуль «Новини», який використовується для оперативного розміщення, редагування та публікації інформаційних повідомлень;

- модуль «Фотогалерея», що дає змогу створювати галереї зображень, групувати їх за категоріями та автоматично масштабувати графічні матеріали;

- модуль «Налаштування», у якому зберігаються параметри функціонування сайту та конфігураційні дані системи управління;

- модуль «Користувачі», призначений для керування обліковими записами, ролями та правами доступу зареєстрованих користувачів;

- модуль «Каталог», який забезпечує додавання, редагування та структурування інформації про об'єкти, товари, послуги або організації в групах різного рівня вкладеності;

- модуль «Опитування», що дає змогу створювати та редагувати анкети, збирати відповіді користувачів і відображати результати у вигляді узагальнених статистичних показників або графічних матеріалів.

Використання системи керування вмістом сайту не потребує встановлення спеціалізованого програмного забезпечення на робочому місці адміністратора, оскільки основні операції виконуються через веббраузер. Інтуїтивно зрозумілий інтерфейс спрощує процес адміністрування, зменшує витрати на підтримку вебресурсу та дає змогу оперативно оновлювати інформацію. Для роботи з такою системою достатньо базових навичок користування мережею Інтернет і загального розуміння структури вебсайту.

Можливості адміністрування CMS охоплюють такі функції:

- створення необмеженої або значної кількості сторінок відповідно до потреб вебресурсу;
- забезпечення чіткої логічної структури сайту та зручної навігації;
- створення описів і ключових слів для динамічних сторінок з метою поліпшення їх індексації пошуковими системами;
- планування дати початку та завершення публікації матеріалів за допомогою календаря;
- обмеження доступу до окремих розділів вебсайту лише для зареєстрованих або авторизованих користувачів;
- завантаження зображень через браузер до внутрішньої бібліотеки для подальшого використання в різних розділах сайту;
- організація опитувань і голосувань як засобів отримання зворотного зв'язку від користувачів;
- підключення додаткових модулів, зокрема блоку останніх новин, лічильника відвідувань, статистики переглядів, форуму або гостьової книги;
- створення декількох форм зворотного зв'язку для різних категорій звернень або контактних осіб;

- зміну порядку відображення інформаційних об'єктів, зокрема новин, запитань, статей і сторінок;
- автоматизований показ випадкової новини або іншого інформаційного матеріалу;
- приймання матеріалів від авторів, користувачів або відповідальних працівників для подальшої модерації та публікації;
- формування ієрархічної структури об'єктів, що може включати секції, розділи, підрозділи та сторінки;
- використання бібліотеки зображень для зберігання та швидкого доступу до графічних файлів;
- керування розсилкою новин та інформаційних повідомлень користувачам;
- ведення архіву опублікованих матеріалів;
- забезпечення можливості друку або надсилання матеріалів користувачам за допомогою електронної пошти;
- використання візуальних редакторів, що спрощують процес підготовки матеріалів до рівня роботи зі звичайним текстовим редактором;
- можливість зміни дизайну вебсайту без повної перебудови його інформаційної структури;
- раціональне використання серверного простору завдяки зберігання даних у базі даних;
- застосування зрозумілих адрес сторінок, адаптованих для ефективнішої індексації пошуковими системами.
- За допомогою системи управління вебсайтом можуть створюватися та підтримуватися різні типи вебресурсів, зокрема:
- корпоративні вебсайти та інформаційні портали;

- інтернет-магазини та електронні каталоги товарів;
- невеликі інформаційні сайти підприємств або окремих проєктів;
- вебсайти некомерційних організацій, освітніх установ та громадських об'єднань;
- персональні сторінки та професійні вебпортфоліо;
- сайти спільнот, тематичних груп і професійних об'єднань;
- онлайн-журнали, електронні видання, новинні ресурси та інші інформаційні платформи.

Висновок до розділу 1

У першому розділі було здійснено аналіз предметної області, визначено особливості функціонування магазину та обґрунтовано необхідність автоматизації процесів, пов'язаних з обліком товарів, опрацюванням замовлень і взаємодією з клієнтами. Розглянуто організаційну структуру підприємства, основні інформаційні потоки та функції учасників торговельного процесу, що дало змогу сформулювати вимоги до майбутньої інформаційної системи. Аналіз існуючих підходів і засобів управління вебресурсами засвідчив доцільність створення системи, адаптованої до потреб конкретного підприємства. Отже, результати дослідження є підґрунтям для подальшого проєктування програмного та інформаційного забезпечення системи автоматизації діяльності магазину.

2 РОЗРОБКА ПРОЕКТНИХ РІШЕНЬ

2.1 Інформаційне забезпечення

2.1.1 Вибір засобів моделювання

Для побудови складної інформаційної системи необхідним етапом є проєктування. В останнє десятиліття в комп'ютерному світі намітилася тенденція проєктування систем візуальними (наочними) моделями.

Головна особливість індустрії сучасних корпоративних програмних комплексів складається в концентрації складності на початкових етапах аналізу вимог і проєктування специфікацій при відносно невисокій складності й трудомісткості наступних етапів. Фактично тут і приходить розуміння того, що буде робити майбутня програма і який чином вона буде функціонувати, щоб задовольнити пред'явленим до неї вимогам. А саме нечіткість і неповнота системних вимог, невирішені питання й помилки, допущені на етапах аналізу й проєктування, породжують на наступних етапах важкі, часто нерозв'язні проблеми й, в остаточному підсумку, приводять до неуспіху всієї роботи в цілому.

Для подолання складностей початкових етапів розробки насамперед і призначений структурний аналіз - метод дослідження, що починається із загального огляду системи й потім деталізується, здобуваючи ієрархічну структуру із все більшим числом рівнів. Для таких методів характерна розбивка на рівні абстракції з обмеженням числа елементів на кожному з рівнів (звичайно від 3 до 6-7); обмежений контекст, що включає лише істотні на кожному рівні деталі; використання строгих формальних правил запису; і як наслідок - послідовне наближення до кінцевого результату.

У цей час відомо порядку 90 різновидів структурного системного аналізу, які можуть бути класифіковані стосовно шкіл (для моделювання програмних систем або систем взагалі), один по одному побудови моделі (декларуючі первинність функціонального або інформаційного моделювання), по типі цільових систем (інформаційні системи або системи реального часу). Незважаючи на такий достаток методів, практично в усіх з них використовуються три групи засобів:

DFD (Data Flow Diagrams) - діаграми потоків даних або SADT-діаграми, що ілюструють функції, які система повинна виконувати;

ERD (Entity-Relationship Diagrams) - діаграми "сутність-зв'язок", що моделюють відносини між даними;

STD (State Transition Diagrams) - діаграми переходів станів, що моделюють залежне від часу поведіння системи (аспекти реального часу).

Крім цих моделей на етапі структурного проектування використовуються техніки структурних карт, призначені для опису відносин між модулями (структурні карти Константайна) і внутрішньої структури модулів (структурні карти Джексона).

Перераховані моделі в сукупності дають повний опис системи незалежно від того, чи є вона існуючої або знову розроблювальної.

2.1.2 Розробка діаграма варіантів використання інформаційної системи магазину продажу мобільного зв'язку

Для зображення статичної моделі системи, уточнення функціональності, виявлення схеми функціональної структури, визначення модулів, які входять до її складу, побудуємо діаграму варіантів використання, яка відображатиме системні функції, системне оточення (діючих осіб або акторів - actors) та зв'язки між ними.

За допомогою моделі варіантів використання відобразимо інтерфейс інформаційної системи з зовнішнім світом, а саме, інформаційні потоки між системою й зовнішніми сутностями, які виникають при виконанні певних функцій. Модель варіантів використання представлена на Рисунку 2.1

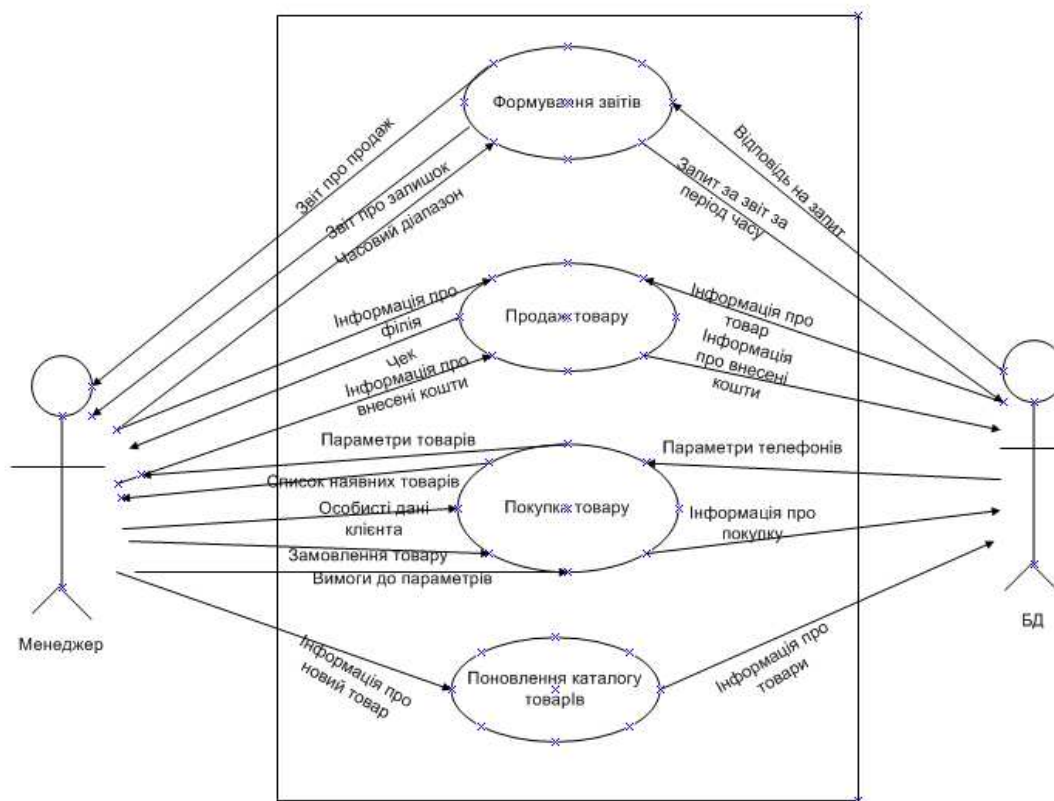


Рисунок 2.1 – Діаграма варіантів використання інформаційної систему магазину мобільного зв'язку.

Розглянемо більш детально кожний варіант використання, надаючи опис потоків подій.

2.1.3 Розробка концептуальної моделі даних

Концептуальна модель - це відображення предметної області, для якої розробляється база даних. Не вдаючись в теорію, відзначимо, що це якась

діаграма з прийнятими позначеннями елементів. Так, всі об'єкти, що позначають речі, позначаються у вигляді прямокутника. Атрибути, що характеризують об'єкт - у вигляді овалу, а зв'язки між об'єктами - ромбами.

Модель сутність-зв'язок (ER-модель) - модель даних, що дозволяє описувати концептуальні схеми предметної області.

ER-модель використовується при високорівневої (концептуальному) проектуванні баз даних. З її допомогою можна виділити ключові сутності і позначити зв'язку, які можуть встановлюватися між цими сутностями.

Під час проектування баз даних відбувається перетворення ER-моделі в конкретну схему бази даних на основі вибраної моделі даних (реляційної, об'єктної, мережевий або ін.)

ER-модель являє собою формальну конструкцію, яка сама по собі не наказує ніяких графічних засобів її візуалізації. В якості стандартної графічної нотації, за допомогою якої можна візуалізувати ER-модель, була запропонована діаграма сутність-зв'язок (ER-діаграма)

Дана нотація була введена Ченом (Chen) і отримала подальший розвиток в роботах Баркера (Barker). Нотація Чена надає багатий набір засобів моделювання даних, включаючи власне ERD, а також діаграми атрибутів і діаграми декомпозиції..

Виходячи з аналізу функціональної моделі системи можна виділити конкретні сутності, ідентифікувати відношення між ними, визначити типи цих відношень. Розглянемо на прикладі відділу з продажів мобільних телефонів.

Можна виділити наступні сутності й атрибути:

Продукт: включає мобільний телефон. Ця сутність має наступні властивості:

- марка – марка мобільного телефону;
- модель – модель мобільного телефону;

- тип корпусу;
- тип телефонна;
- тип батареї;

Мобільний телефон: має зв'язок з сутністю Параметри, Відділення, Замовлення. Ця сутність має наступні властивості:

- ціна – ціна за 1 одиницю товару;
- кількість – кількість товару;

Відділення: додає мобільний телефон. Має зв'язок з сутністю мобільний телефон. Ця сутність має наступні властивості:

- назва – назва відділення;
- адрес – місце знаходження відділення;

Замовлення: має мобільний телефон. Має зв'язок з сутністю мобільний телефон. Ця сутність має наступні властивості:

- дата – день, число, місяць коли було прийнято замовлення;
- кількість – кількість замовлення;
- ціна – на яку суму було прийнято замовлення;

Параметри: має мобільний телефон. Має зв'язок з сутністю мобільний телефон. Ця сутність має наступні властивості:

- параметр – має значення;
- значення – має параметр.

Побудуємо концептуальну модель даних системи. Найбільш розповсюдженим засобом представлення концептуальної моделі даних є діаграми "сутність-зв'язок" (ERD). На рисунку 2.3 представлена ER діаграма у нотації Чена.

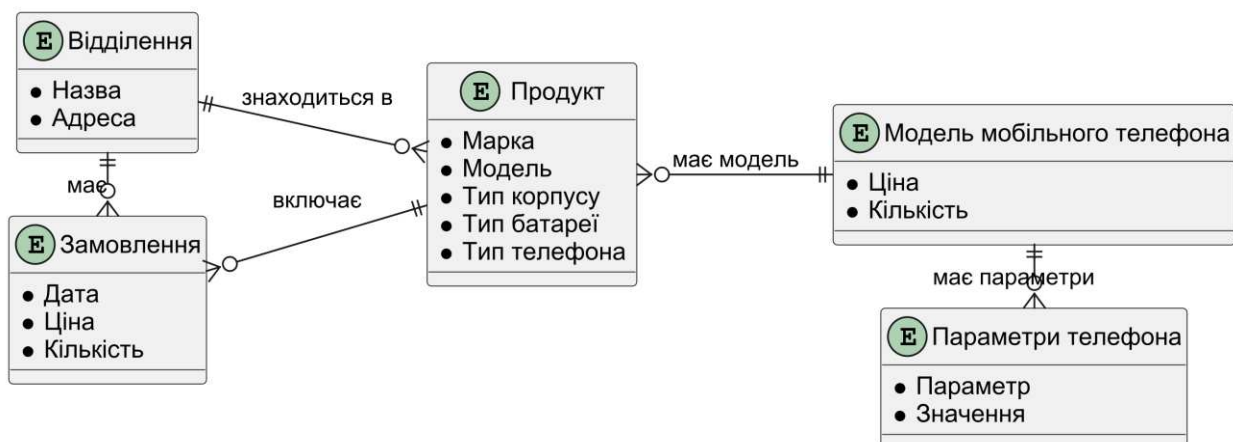


Рисунок 2.2 – Концептуальна модель даних відділу мобільного зв'язку магазину побутової техніки.

2.1.4 Логічна модель даних

Базою для побудови логічної моделі даних є концептуальна модель. Логічна модель є більш детальним представленням даних. В ній сутності трансформуються в таблиці даних, одна сутність може представлятися декількома таблицями, атрибути сутностей стають полями таблиць, уточнюються відношення між таблицями, виділяються первинні ключі.

Логічні моделі структур даних повинні забезпечувати можливість реалізувати інформаційні моделі, представлені у попередньому підрозділі, в певній системі програмування. У даній роботі буде використовуватись реляційна модель даних, і відповідна логічна модель програмного забезпечення автоматизації продажу мобільних телефонів з деталізацією атрибутів показана на рисунку 2.3.

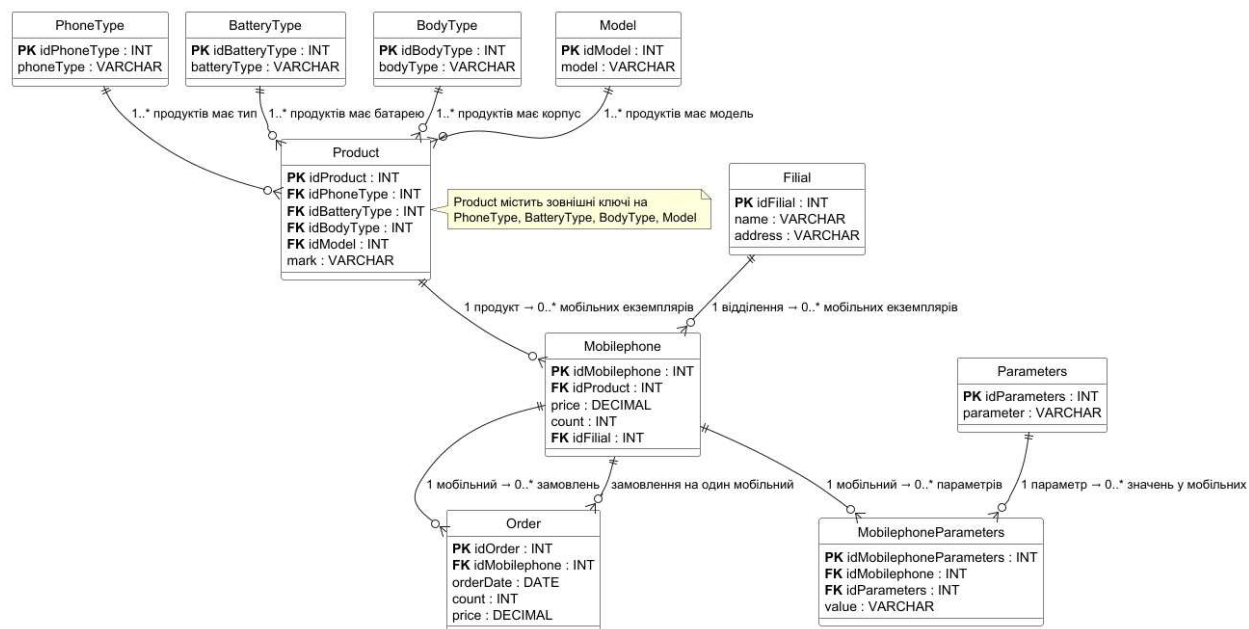


Рисунок 2.3 – Логічна модель даних

2.1.5 Розробка фізичної моделі даних

Фізична модель даних будується методом денормалізації та типізації логічної моделі даних. Розробимо фізичну структуру кожної таблиці, що входить у базу даних.

Фізична структура таблиць, які входять до БД інформаційного забезпечення представлена на рисунку 2.4.

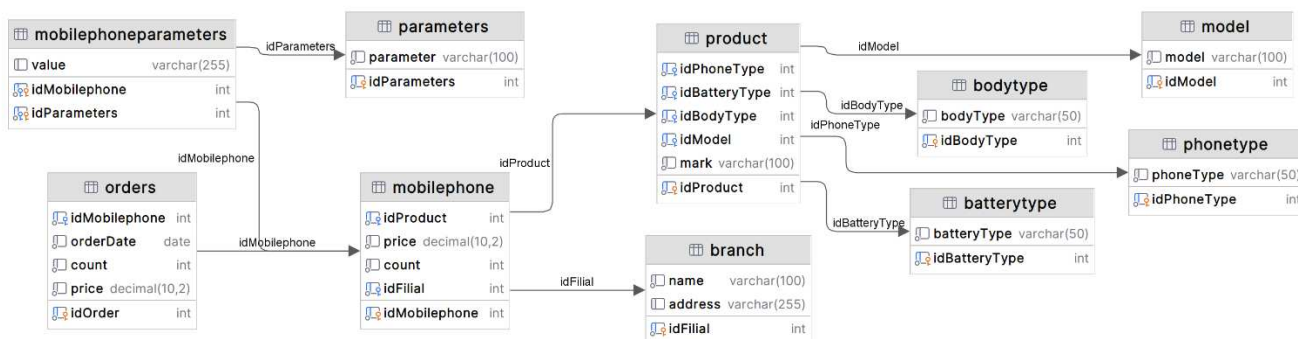


Рисунок 2.4 – Фізична модель даних

2.2 Програмне забезпечення

2.2.1 Вибір мови програмування

З огляду на результати аналізу середовища виконання та функціональні вимоги до програмного забезпечення інформаційної системи магазину, вибір мови програмування має ґрунтуватися на її придатності до створення надійних, масштабованих і супроводжуваних застосунків. У межах розроблення АРМ менеджера до мови програмування доцільно висунути такі вимоги:

- платформна незалежність і можливість виконання програм у різних операційних середовищах;
- підтримка мережевої взаємодії та побудови клієнт-серверних або веборієнтованих рішень;
- наявність стандартизованих засобів роботи з базами даних;
- гнучкість архітектури та можливість подальшого розвитку програмного продукту.

З урахуванням зазначених вимог для реалізації поставленого завдання обрано мову програмування Java, яка належить до поширених технологій створення корпоративних, серверних, настільних і веборієнтованих програмних систем.

Java є об'єктно-орієнтованою мовою програмування загального призначення та одночасно програмною платформою, що функціонує на основі віртуальної машини Java. Її ключова концепція полягає у можливості компіляції вихідного коду в проміжний байт-код, який виконується у середовищі JVM. Такий підхід забезпечує високий рівень переносимості

програмного забезпечення між різними апаратними й операційними платформами.

- об'єктно-орієнтована модель програмування з підтримкою інкапсуляції, наслідування та поліморфізму;
- платформна незалежність завдяки виконанню байт-коду у віртуальній машині Java;
- розвинена стандартна бібліотека для роботи з колекціями, потоками введення-виведення, мережевими протоколами, багатопотоковістю та базами даних;
- наявність сучасних мовних засобів, зокрема лямбда-виразів, Stream API, записів, узгодження із шаблонами, модульної системи та віртуальних потоків;
- стала екосистема інструментів, фреймворків і бібліотек, що спрощують розроблення, тестування, розгортання та супровід програмних продуктів.

Сучасна Java широко застосовується для створення серверних застосунків, корпоративних інформаційних систем, вебсервісів, мікросервісної архітектури, систем електронної комерції та прикладних програм, що працюють з базами даних. Для взаємодії з реляційними СУБД використовується стандартний механізм JDBC, а для мережевої взаємодії — засоби роботи з HTTP, TCP/IP та іншими протоколами. Завдяки цьому Java є придатною для реалізації інформаційної системи магазину, у якій необхідні стабільна робота з даними, оброблення запитів користувачів і можливість подальшого масштабування.

Компілятор Java формує байт-код, який не прив'язаний до конкретної операційної системи. Виконання такого коду забезпечується віртуальною машиною Java, що може інтерпретувати його або перетворювати у машинний код під час виконання із застосуванням JIT-компіляції. Це дає змогу

використовувати один і той самий програмний код на різних платформах за умови наявності відповідного середовища виконання.

Важливо враховувати, що Java активно розвивається за регулярним циклом випусків, а для промислового використання доцільно орієнтуватися на версії з довгостроковою підтримкою. До таких належать Java 17, Java 21 та Java 25, причому Java 21 стала важливим етапом модернізації платформи завдяки стабілізації віртуальних потоків, удосконаленню шаблонів у конструкції switch і розвитку засобів опису даних. Java 25 є актуальною LTS-версією, однак вибір конкретної версії має визначатися сумісністю з використаними бібліотеками, середовищем розгортання та вимогами до підтримки.

Для розроблення програмного забезпечення інформаційної системи доцільним є використання сучасної LTS-версії Java, зокрема Java 21 або Java 25, оскільки вони забезпечують тривалий період підтримки, сучасні мовні можливості, належний рівень продуктивності та сумісність із поширеними інструментами розроблення. Такий вибір підвищує надійність системи та зменшує ризики, пов'язані з використанням застарілих версій платформи.

- підтримка сучасних засобів опису даних і зменшення шаблонного коду завдяки записам та удосконаленим конструкціям мови;

- можливість ефективної організації паралельного виконання та оброблення великої кількості операцій введення-виведення за допомогою віртуальних потоків;

- покращені механізми безпеки, продуктивності та керування пам'яттю у сучасних реалізаціях JVM;

- широка підтримка з боку середовищ розроблення, систем збирання, фреймворків і драйверів для роботи з базами даних.

2.2.2 Методи та засоби розробки

Для побудови, аналізу та документування моделей даних у межах розроблення інформаційної системи доцільно використовувати сучасні CASE-засоби, що підтримують концептуальне, логічне та фізичне моделювання баз даних. З огляду на актуальний стан програмних продуктів замість застарілої версії ERwin 7.3 доцільно орієнтуватися на сучасний продукт erwin Data Modeler by Quest, зокрема версії лінійки 15.x.

erwin Data Modeler є спеціалізованим програмним засобом для проєктування, візуалізації, аналізу та супроводу моделей даних. Він забезпечує створення моделей різних рівнів деталізації, підтримує пряму та зворотну інженерію баз даних, формування схем, порівняння моделей і баз даних, а також документування структури інформаційних ресурсів.

У сучасних версіях erwin Data Modeler підтримуються реляційні СУБД, хмарні платформи, сховища даних, окремі NoSQL-підходи, засоби спільної роботи, керування метаданими та інтеграція з інструментами управління даними. Це дає змогу застосовувати продукт не лише для побудови ER-діаграм, а й для підтримки повного життєвого циклу моделі даних.

Використання такого інструмента є обґрунтованим під час проєктування бази даних інформаційної системи магазину, оскільки він надає графічний інтерфейс для побудови моделей, засоби опису сутностей, атрибутів і зв'язків, механізми нормалізації структури даних, а також можливості генерації фізичної схеми бази даних для обраної СУБД.

Процес побудови інформаційної моделі за допомогою сучасного CASE-засобу охоплює такі основні етапи:

- визначення предметної області, сутностей та їхніх ключових характеристик;
- встановлення зв'язків між сутностями та визначення їхньої кардинальності;
- визначення первинних, зовнішніх та альтернативних ключів;
- опис атрибутів, типів даних, обмежень цілісності та правил найменування;
- нормалізація логічної моделі та усунення надлишковості даних;
- перетворення логічної моделі у фізичну модель з урахуванням особливостей обраної СУБД;
- генерація SQL-скриптів, документації та, за потреби, синхронізація моделі з наявною базою даних.

Візуальна модель даних, створена в erwin Data Modeler, може використовуватися для аналізу структури майбутньої бази даних, узгодження проєктних рішень із зацікавленими сторонами, формування технічної документації та подальшого супроводу системи. Підтримка сучасних платформ і механізмів інтеграції підвищує придатність цього інструмента для розроблення інформаційних систем, орієнтованих на довгострокову експлуатацію.

Під час подальшого проєктування програмного забезпечення доцільно застосовувати принципи об'єктно-орієнтованого аналізу та проєктування, зокрема інкапсуляцію, наслідування, поліморфізм і поділ відповідальностей між класами. Такий підхід сприяє зменшенню складності програмного коду, підвищенню його повторного використання, спрощенню тестування та зниженню ймовірності помилок.

2.2.3 Структура програмного забезпечення.

Вхідними даними для цього етапу проєктування є концептуальна та логічна моделі предметної області, а результатом — специфікація основних програмних компонентів, класів, інтерфейсів і зв'язків між ними. Для формалізації структури програмного забезпечення доцільно використовувати діаграми класів UML, які дають змогу відобразити склад системи, атрибути об'єктів, операції класів та характер взаємодії між окремими елементами.

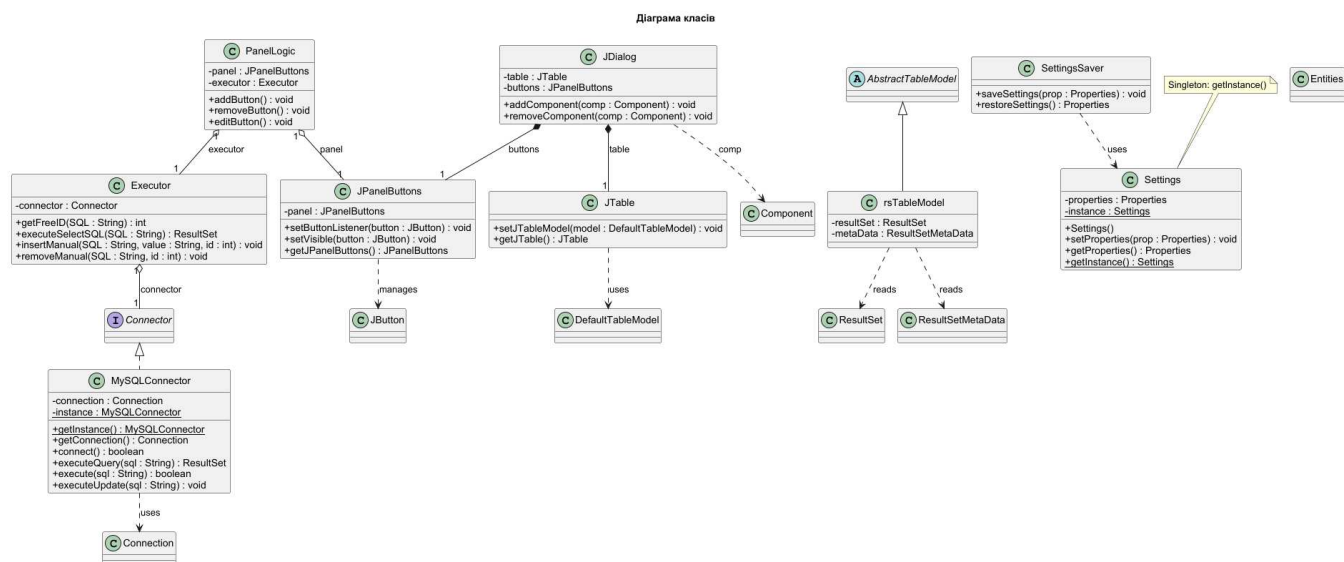


Рисунок 2.5 – Діаграма класів системи

Сформульовані користувацькі вимоги розглядаються як функціональні та нефункціональні обмеження, що визначають архітектуру програмного забезпечення, склад його модулів, правила доступу до даних і вимоги до надійності, масштабованості та супроводжуваності системи.

Укажемо відповідальність кожного з приведених на діаграмі класів:

Connector (Інтерфейс) – для використання різних СУБД;

MySQLConnector– реалізація інтерфейса, та підключення до БД;

Entities - інтерфейс з запитами до БД;

AbstractTableModel– абстрактна модель таблиці;

rsTableModel – успадкування і перевизначення методів абстрактної моделі таблиці;

JTable (таблиця з методами) – встановлення моделі отримання таблиці;

JDialog – вікно з таблицею і набір кнопок;

JPanelButtons – панель з кнопками;

PanelLogic (логіка програми) – методи які виконуються при натисканні на кнопки;

Executor - клас який виконує запити до БД;

SettingsSaver (зчитування) – запис налаштувань в файл;

Settings – налаштування підключення до БД (зміна, отримання).

2.2.4 Розгортання програмного забезпечення

Діаграма розгортання необхідна для візуалізації елементів та компонентів системи. Розробка діаграми розгортання являється останнім етапом специфікації моделі системи. При розробці даної діаграми визначається розподіл компонентів системи по її фізичним вузлам.

Розроблену діаграму розгортання представлено на рисунку 2.6



Рисунок 2.6 – Діаграма розгортання системи

Висновок до розділу 2

У другому розділі було сформовано основні проєктні рішення щодо інформаційного та програмного забезпечення системи автоматизації

діяльності магазину мобільного зв'язку. Розроблено концептуальну, логічну та фізичну моделі даних, що визначають структуру майбутньої бази даних і забезпечують узгодженість інформаційних потоків у межах системи. Обґрунтовано вибір сучасних засобів моделювання, мови програмування та підходів до побудови програмної архітектури, зокрема використання UML-діаграм для опису функціональності, класів і розгортання системи. Отже, отримані проєктні рішення створюють основу для подальшої реалізації, тестування та впровадження програмного продукту в практичну діяльність підприємства.

3 РЕАЛІЗАЦІЯ ПРОЕКТУ АВТОМАТИЗОВАНОЇ СИСТЕМИ

У межах дипломної роботи було реалізовано автоматизоване робоче місце менеджера магазину мобільного зв'язку, призначене для підвищення ефективності виконання основних торговельних, облікових та інформаційно-аналітичних операцій. Розроблена система орієнтована на автоматизацію процесів ведення товарного каталогу, обліку наявності продукції, фіксації продажів, опрацювання замовлень, формування звітних даних і підтримання актуальності інформації, що використовується працівниками підприємства під час прийняття управлінських рішень.

Необхідність створення такого програмного забезпечення зумовлена тим, що діяльність магазину характеризується постійним оновленням асортименту, зміною цін, надходженням нових товарних позицій, виконанням замовлень клієнтів і потребою в оперативному отриманні достовірних відомостей про залишки продукції. За умов ручного або частково автоматизованого ведення обліку зростає ймовірність виникнення помилок, дублювання даних, несвоєчасного оновлення інформації та ускладнення контролю за рухом товарів. Тому розроблена інформаційна система спрямована на упорядкування інформаційних потоків і забезпечення централізованого доступу до даних, необхідних для виконання посадових обов'язків менеджера.

Функціональні можливості програмного продукту сформовано відповідно до вимог, визначених у технічному завданні, наведеному в додатку А. Під час реалізації системи було враховано необхідність забезпечення зручності роботи користувача, цілісності даних, логічної послідовності виконання операцій і можливості подальшого розширення функціоналу. Основними функціями системи є ведення довідника товарів, додавання нових

товарних позицій, редагування відомостей про продукцію, внесення додаткових технічних характеристик, реєстрація продажів, пошук інформації в загальній базі даних, а також формування звітів про реалізовані та замовлені товари.

Однією з ключових функцій системи є додавання товарної позиції до електронного каталогу. У процесі виконання цієї операції користувач вводить основні відомості про товар, зокрема його найменування, виробника, модель, ціну, кількість на складі та інші характеристики, необхідні для подальшої ідентифікації й обліку. Такий підхід дає змогу підтримувати актуальний перелік продукції та забезпечує можливість швидкого доступу до інформації про конкретний товар. У разі зміни вартості, кількості або опису продукції передбачено редагування товарної позиції, що забезпечує відповідність даних фактичному стану асортименту магазину.

Окрему роль у роботі системи відіграє модуль внесення додаткової інформації про товар. Для магазинів мобільного зв'язку важливими є не лише загальні відомості про продукцію, а й технічні параметри, що впливають на вибір покупця: тип пристрою, обсяг пам'яті, характеристики екрана, ємність акумулятора, підтримувані стандарти зв'язку, комплектація та інші споживчі властивості. Наявність таких даних у системі спрощує консультаційну роботу менеджера, підвищує якість обслуговування клієнтів і дає змогу швидше порівнювати різні товарні позиції між собою.

Функція внесення інформації про продаж товару забезпечує фіксацію факту реалізації продукції та оновлення відповідних показників у базі даних. Після виконання продажу система може зменшувати кількість товару на складі, зберігати відомості про дату операції, кількість реалізованих одиниць і суму продажу. Такий механізм є важливим для контролю товарних залишків, аналізу попиту та своєчасного прийняття рішень щодо необхідності поповнення

асортименту. Крім того, накопичені дані про продажі можуть використовуватися для формування аналітичних звітів і визначення найбільш затребуваних товарних позицій.

Пошук інформації в загальній базі товарів реалізовано з метою прискорення доступу до необхідних відомостей. Користувач може здійснювати пошук за найменуванням, кодом, моделлю, виробником або іншими ознаками, передбаченими структурою бази даних. Це особливо важливо для підприємств, що мають широкий асортимент або декілька торговельних точок, оскільки швидке отримання інформації про наявність товару дає змогу оперативно відповідати на запити клієнтів і підвищувати рівень сервісу.

Формування звітів про продані товари є одним із важливих інструментів інформаційної підтримки управління магазином. Звітні форми дають змогу узагальнювати відомості про обсяги реалізації, аналізувати динаміку продажів, оцінювати ефективність товарного асортименту та визначати позиції, що потребують додаткової уваги з боку менеджера. Використання автоматизованого формування звітів скорочує час підготовки управлінської інформації та зменшує ризик арифметичних або логічних помилок, характерних для ручного опрацювання даних.

Усі функції програмного забезпечення реалізовано у відповідних модулях системи, кожен з яких виконує окрему логічно завершену групу операцій. Такий підхід відповідає принципам модульного проєктування та забезпечує кращу структурованість програмного коду. Розподіл функціональності між окремими компонентами спрощує супровід системи, дає змогу швидше виявляти й усувати помилки, а також створює передумови для подальшого розширення програмного продукту без істотної зміни його базової архітектури.

Реалізація системи ґрунтується на використанні об'єктно-орієнтованого підходу, що дає змогу представити основні елементи предметної області у вигляді класів, об'єктів та їхніх взаємозв'язків. Такий підхід є доцільним для систем, які працюють із взаємопов'язаними сутностями, зокрема товарами, замовленнями, продажами, користувачами та довідковими даними. Використання класів забезпечує інкапсуляцію даних і операцій над ними, підвищує повторне використання програмного коду та сприяє зниженню складності розроблення.

Важливим елементом реалізації є організація взаємодії програмного забезпечення з базою даних. База даних призначена для централізованого зберігання інформації про товари, їхні характеристики, залишки, операції продажу та інші облікові відомості. Наявність єдиного джерела даних забезпечує узгодженість інформації, усуває дублювання записів і створює умови для оперативного оновлення відомостей. Для користувача це означає можливість отримання актуальної інформації без необхідності звернення до кількох розрізнених джерел.

Інтерфейс користувача системи має забезпечувати зрозумілу послідовність виконання операцій і мінімізувати кількість дій, необхідних для отримання результату. Для цього функції, що найчастіше використовуються менеджером, мають бути логічно згруповані, а форми введення даних — містити лише ті поля, які є необхідними для виконання конкретної операції. Така організація інтерфейсу сприяє зменшенню навантаження на користувача, підвищує швидкість роботи з програмою та знижує ймовірність помилкового введення даних.

З практичного погляду впровадження автоматизованого робочого місця менеджера дає змогу підвищити якість інформаційного забезпечення торговельної діяльності. Менеджер отримує можливість швидше

опрацьовувати запити клієнтів, перевіряти наявність товарів, фіксувати продажі, оновлювати каталог і формувати звітні матеріали. У результаті скорочується час виконання типових операцій, підвищується точність облікових даних і поліпшується контроль за рухом товарних позицій.

Результатом роботи розробленої підсистеми є надання користувачеві необхідної інформації у зручній для сприйняття формі. Залежно від виконуваної операції користувач може отримати перелік товарів, відомості про конкретну товарну позицію, інформацію про залишки, дані щодо продажів або сформований звіт. Отже, система виконує не лише облікову, а й інформаційно-аналітичну функцію, оскільки забезпечує підготовку даних, необхідних для прийняття обґрунтованих управлінських рішень.

Під час експлуатації програмного забезпечення користувачеві доцільно керуватися супровідною документацією, зокрема документами «Керівництво користувача» та «Технічний опис», наведеними відповідно в додатках Б і В. Керівництво користувача містить опис основних операцій, порядок роботи з інтерфейсом і рекомендації щодо виконання типових дій у системі. Технічний опис, у свою чергу, розкриває особливості структури програмного забезпечення, організації його компонентів, взаємодії з базою даних і умов розгортання. Наявність таких документів забезпечує зрозумілість процесу використання системи та спрощує її подальший супровід.

Таким чином, реалізована автоматизована система відповідає поставленим функціональним вимогам і забезпечує комплексну підтримку діяльності менеджера магазину мобільного зв'язку. Її використання сприяє підвищенню оперативності оброблення інформації, зменшенню трудомісткості рутинних операцій, покращенню якості обліку та формуванню більш надійної інформаційної основи для управління торговельними процесами.

Висновок до розділу 3

У третьому розділі було розглянуто практичну реалізацію автоматизованої системи, призначеної для підтримки діяльності менеджера магазину мобільного зв'язку. У процесі реалізації програмного продукту забезпечено виконання основних функцій, пов'язаних із веденням товарного каталогу, обліком наявності продукції, реєстрацією продажів, пошуком інформації в базі даних і формуванням звітних матеріалів. Особливу увагу приділено модульній організації системи, взаємодії програмного забезпечення з базою даних, зручності користувацького інтерфейсу та забезпеченню цілісності інформації. Розроблена система дає змогу скоротити час виконання типових операцій, підвищити точність облікових даних і створити надійну інформаційну основу для прийняття управлінських рішень. Отже, результати реалізації підтверджують практичну доцільність запропонованих проєктних рішень і можливість подальшого використання програмного продукту в діяльності підприємства.

4 ОХОРОНА ПРАЦІ

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людини в процесі праці (ст.1 Закону "Про охорону праці"). Як правовий інститут, охорона праці – це чималий комплекс правових норм.

Ст. 4 Закону "Про охорону праці" формулює принципи державної політики у сфері охорони праці. Ці принципи в силу вказаного їх визначення, слід кваліфікувати і як принципи інституту охорони праці, які підлягають застосуванню при відсутності відповідних конкретних правових норм, і як такі, що визначають напрямки подальшого розвитку інституту охорони праці, і як обов'язки держави, які вона взяла на себе і має додержувати при здійсненні своєї політики у майбутньому. З урахуванням викладеного, принципи, зазначені в ст. 4 Закону "Про охорону праці", мають певне регулятивне значення. У названій статті закріплені такі принципи:

1) пріоритет життя і здоров'я працівників відносно результатів виробничої діяльності підприємства, повна відповідальність власника за створення безпечних і нешкідливих умов праці;

2) комплексне вирішення завдань охорони праці на засадах національних програм з цих питань, а також з урахуванням інших напрямків економічної і соціальної політики, досягнень в галузі науки, техніки та охорони здоров'я;

3) соціальний захист працівників, повне відшкодування шкоди особам, потерпілим від нещасних випадків на виробництві та професійних захворювань;

4) установлення єдиних нормативів з охорони праці для всіх підприємств незалежно від форм власності і видів їх діяльності;

5) використання економічних методів управління охороною праці, проведення політики пільгового оподаткування, що сприяє створенню безпечних та нешкідливих умов праці, участь держави в фінансуванні заходів з охорони праці;

6) здійснення навчання населення, професійної підготовки та підвищення кваліфікації працівників з питань охорони праці;

7) забезпечення координації діяльності державних органів, установ, організацій та об'єднань громадян, що вирішують різні проблеми охорони здоров'я, гігієни та безпеки праці, а також співробітництво.

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці.

Власник або уповноважений ним орган повинен впроваджувати сучасні засоби техніки безпеки, які запобігають виробничому травматизмові, і забезпечувати санітарно-гігієнічні умови, що запобігають виникненню професійних захворювань працівників.

На власника або уповноважений ним орган покладається систематичне проведення інструктажу (навчання) працівників з питань охорони праці, протипожежної охорони.

Охорона праці – один з центральних інститутів трудового права. Він має виключно практичне значення. Недодержання вимог охорони праці створює небезпеку для здоров'я і життя працівників. У свою чергу і ті, кого законодавець називає власником або уповноваженим ним органом, несуть сувору, у тому числі й кримінальну, відповідальність за порушення правил охорони праці.

Підвищення технічного рівня виробництва нерозривно зв'язано із широким використанням різних видів електроніки, телебачення, оптичних квантових генераторів, електронно-обчислювальних машин. Усі ці пристрої є джерелами електромагнітних випромінювань, що зараз прийнято називати єдиним терміном - неіонізуючі випромінювання. Коло осіб, що піддаються впливові таких випромінювань, увесь час розширюється.

Швидкий темп змін в умовах праці великого контингенту трудящих приводить до посилення впливу на працюючу людину нових раніше невідомих або мало вивчених небезпечних і шкідливих виробничих факторів фізичного, біологічного й особливо психофізіологічного характеру. У зв'язку з цим необхідно удосконалювати роботи з охорони праці, упроваджувати нові ефективні засоби забезпечення безпеки праці.

4.1 Аналіз небезпечних і шкідливих факторів в офісному приміщенні

Небезпечні і шкідливі виробничі фактори. Класифікація", приводиться класифікація елементів умов праці, що виступають у ролі небезпечних і шкідливих виробничих факторів. Вони підрозділяються на 4 групи: фізичні, хімічні, біологічні і психофізіологічні.

Експлуатація ЕОМ супроводжується виникненням ряду шкідливих і небезпечних виробничих факторів фізичного і психофізіологічного характеру. Працівники, що обслуговують ЕОМ, піддаються під час роботи впливові наступних небезпечних і шкідливих факторів:

- автоматизація обліку руху товару;
- фізичних: підвищення рівня шуму, статичної електрики, електромагнітних випромінювань, підвищена напруженість електричних і магнітних полів, недостатня освітленість робочого місця, підвищена

яскравість світла;

- психофізіологічних: фізичні і нервово-психічні перевантаження (розумова перенапруга, монотонність праці, емоційні перевантаження).

Санітарними нормами і правилами встановлені наступні припустимі значення:

- рівень шуму: 50 дБ;
- гранично припустима напруженість електростатичного поля на

робочому

місці при тривалості впливу на людину до 1 години: 60 кВ/м.

Інтенсивність впливу неіонізуючих магнітних полів на обслуговуючий персонал (оператори, програмісти) залежить від потужності джерел випромінювання, режиму роботи, конструктивної особливості випромінюючих пристроїв, технічного стану устаткування, розташування робочого місця й ефективності захисних заходів. Гранично припустима напруженість електричної складової електромагнітного поля складає 10 В/м, магнітної складової 0.3 А/м. Тривалий вплив неіонізуючих електромагнітних випромінювань на організм людини з інтенсивністю перевищуючі гігієнічні нормативи приводить до функціональних змін з боку нервової, серцево-судинної, нейрогуморальної й іншої систем організму.

Фізичні перевантаження найчастіше торкаються зорової системи або кістково-м'язову систему (плечовий суглоб, шия, верхня частина спини). У цілому користувачі ЕОМ не зіштовхуються з якимсь особливим психічним стресом або емоційними проблемами, дослідження показують одиничні випадки дуже напружений, зухвалий стрес, роботи на ЕОМ. Проте, рекомендується вживати заходів до запобігання надмірного дискомфорту і стомлення працюючих на ЕОМ. У більшості користувачів ЕОМ виникає кістково-м'язовий дискомфорт через статичну напругу м'язів, незручної пози

або часто повторюваних одноманітних рухів, імовірність виникнення стійких порушень дуже мала, однак монотонні рухи при роботі на ЕОМ можуть викликати хронічні захворювання.

Важливим фактором надійної роботи засобів ВТ і високої працездатності обслуговуючого персоналу є кліматичні умови. Під кліматичними умовами виробничого середовища відповідно до ДСТ 12.1.005-76 розуміють сполучення температури, відносній вологості і швидкості руху повітря. Ці мікрокліматичні фактори як кожний окремо, так і в різних сполученнях впливають на функціональну діяльність людини, його самопочуття і здоров'я і на надійність роботи засобів ВТ.

- Температура повітря є одним з основних параметрів, що характеризують тепловий стан мікроклімату. Основними джерелами теплоти в приміщеннях є ЕОМ і допоміжне устаткування, прилади освітлення, що обслуговує персонал. Потрібно враховувати і зовнішні джерела надходження теплоти (теплоту, що надходить через вікна від сонячної радіації і приплив теплоти через непрозорі конструкції, що обгороджують).

На роботу устаткування великий вплив робить відносна вологість повітря. При вологості повітря до 40% підвищується знос магнітних голівок, виходить з ладу ізоляція проводів. При вологості повітря більш 75-80% знижується опір ізоляції, змінюються робочі характеристики елементів ЕОМ, зростає інтенсивність їхніх відмовлень.

Швидкість руху повітря - вектор усередненої швидкості переміщення повітряних потоків під дією різних сил, що спонукують, також впливає на роботу друкувальних пристроїв.

Атмосферний тиск у приміщеннях також нормується. При зниженому тиску повітря погіршується відвід тепла від елементів ЕОМ, знижуються

ізоляційні властивості повітря. Перераховане вище обумовлює підвищені вимоги до кліматичних умов середовища в приміщеннях з ЕОМ.

Оптимальні і припустимі норми мікроклімату:

- температура повітря 18-25 °С;
- відносна вологість повітря 40-70%;
- швидкість руху повітря не більш 0.3 м/с;
- атмосферний тиск 1013.25 + 266 кпа.

Природне і штучне освітлення в приміщеннях регламентується СНІП 23-05-95 у залежності від характеру зорової роботи, системи і виду освітлення, тла, контрасту об'єкта з тлом. Характеристика зорової роботи визначається найменшим розміром об'єкта розрізнення.

Штучне освітлення нормується кількісними і якісними показниками (показниками засліпленості і дискомфорту, коефіцієнтом пульсації освітленості). Природне освітлення характеризується тим, що створювана освітленість змінюється в залежності від часу доби, року, метеорологічних умов. Тому як критерій оцінки природного освітлення прийнята відносна величина коефіцієнт природної освітленості.

Робота комп'ютерів і периферійних пристроїв можлива завдяки електричному струмові, до джерел якого вони підключаються, тому існує небезпека ураження користувача струмом. Щоб уникнути нещасних випадків варто строго дотримуватися правил електробезпеки.

Небезпека електричного струму на відміну від інших небезпек збільшується тим, що людина не в змозі без спеціальних приладів виявити напругу дистанційно, а також швидкоплинністю поразки – небезпека виявляється, коли людина вже уражена. Ураження струмом приводить до різних порушень в організмі, викликаючи як місцеву поразку тканин і органів, так і загальне ураження організму.

Людина відчуває дратівну дію перемінного струму промислової частоти силою 0,6 – 1,5 мА і постійного струму 5 – 7 мА. Ці струми не представляють серйозної небезпеки для організму людини, тому що при такій силі струму можливе самостійне звільнення людини від контакту зі струмоведучими частинами. Для перемінного струму промислової частоти сила невідпускаючого струму знаходиться в межах 6 – 20 мА. Постійний струм не викликає невідпускаючого ефекту, але приводить до сильних болючих відчуттів, сила такого струму 15 – 80 мА і більше.

4.2 Розробка заходів, щодо зменшення впливу небезпечних і шкідливих факторів

1) Заходи щодо зниження впливу електро-магнітного випромінювання

Для зниження впливу електро-магнітного випромінювання треба застосовувати ЖК монітори. Монітори, які відповідають останнім стандартам ТСО'99 та MPR II є найбільш безпечними для здоров'я людини. Вони визначають максимально припустимі величини випромінювання магнітного й електричного полів, а також методи їхнього виміру. MPRII базується на концепції про те, що люди живуть і працюють у місцях, де вже є магнітні й електричні поля, тому пристрою, які ми використаємо, такі, як монітор для комп'ютера, не повинні створювати електричні й магнітні поля, більші ніж ті, які вже існують. Помітимо, що стандарти ТСО вимагають зниження випромінювань електричних і магнітних полів від пристроїв настільки, наскільки це технічно можливо, поза залежністю від електричних і магнітних

полів, що вже існують навколо нас. Втім, ми вже відзначали, що стандарти ТСО жорсткіше, ніж MPRII.

2) Заходи щодо зниження величини виникаючих зарядів статичної електрики

Для зниження величини виникаючих зарядів статичної електрики покриття технологічних підлог варто виконувати з одношарового полівінілхлоридного антистатичного лінолеуму. Іншим методом захисту є нейтралізація заряду статичної електрики іонізованим газом. У промисловості широко застосовуються радіоактивні нейтралізатори. До загальних мір захисту від статичної електрики можна віднести загальне і місцеве зволоження повітря.

4.3 Розробка заходів щодо забезпечення сприятливих умов праці при роботі з персональним комп'ютером

Сьогодні фахівці в області ергономіки вже зрозуміли, що не можна знайти ідеальне положення, у якому можна перебувати і працювати протягом усього робочого дня. Для більшості людей комфортабельним робочим місцем повинне бути таке, котре можна пристосувати не менш чим для двох позицій, при цьому положення крісла, монітора повинні щораз відповідати виконуваній роботі і звичкам. Багато хто вважають, що для роботи на комп'ютері більше підходять вертикальне і злегка похиле положення. Можливо, щоб крісло було злегка нахилене вперед.

Дисплей. Положення тіла звичайно відповідає напрямку погляду; дисплеї, розташовані занадто низько чи під неправильним кутом, є основними причинами сутулості. Відстань від дисплея до очей може варіюватися в

залежності від характеру виконуваної роботи - 40-70 см. При роботі з текстом відстань від екрана до очей повинна лише небагато перевищувати відстань між книгою й очима і складати 40-45 див. Необхідно давати відпочинок очам і час від часу їх просто закривати. Не можна допускати, щоб очі увесь час були сфокусовані на одній відстані, працюючи з текстом, рекомендується встановлювати великий шрифт. Дуже важливо, щоб екран монітора не мерехтів. Частота регенерації зображення повинна бути не менш 72 Гц, тому що при більш низькій частоті помітне мерехтіння, хоча люди з особливо чутливим зором можуть помітити його і при більш високій частоті. Їм доведеться підібрати графічну плату і монітор з частотою регенерації 85 Гц і вище.

Рекомендується встановлювати на екран монітора спеціальні скляні поляризаційні фільтри з заземленням. Необхідно уникати того, щоб термінал був звернений екраном убік вікна, оскільки інтенсивна освітленість області зору може затопити потоками світла очі і розмити зображення на сітківці. Якщо приходить сидіти поруч з вікном, то треба розташуватися під прямим кутом до нього, причому екран дисплея повинний бути перпендикулярний шибці - цим виключаються відблиски на екрані.

Для зниження впливу низькочастотного електромагнітного випромінювання монітора на ЕЛТ рекомендується вибирати монітор, що задовольняє стандарту MPR II, чи ТСО 92-95. У протилежному випадку необхідно установити на екран захисний фільтр, найбільш сучасні моделі, затримують до 99 % електромагнітного випромінювання.

Форма спинки крісла повинна повторювати форму спини працюючого. Крісло необхідно установити на такій висоті, щоб не почувався тиск на куприк (крісло розташоване занадто низько) на стегна (крісло розташоване занадто високо). Фахівці з ергономіки, вважали, що кут між стегнами і хребтом

повинний складати 90°, однак недавно проведені дослідження показали, що більшість людей переважно сидять трохи відкинувшись.

Клавіатуру необхідно установити так, щоб не треба було до неї тягтися і нахилятися вперед. При зміні положення тіла (наприклад, з вертикального на похиле) обов'язково треба перемінити положення клавіатури і дисплея. Може виявитися корисною регульована підставка клавіатури, завдяки якій можна без усякої напруги працювати з маніпулятором "миша". Можна поставити клавіатуру і на коліна. Руки при роботі повинні бути прямими в зап'ястях і зігнуті в ліктях приблизно під прямим кутом. Пальці також повинні бути злегка зігнуті. Удари по клавішах не повинні бути занадто сильними.

Зручна висота столу особливо важлива в тому випадку, коли на ньому розташовується клавіатура. Якщо в клавіатури немає підставки, а висоту столу не можна змінити (і він занадто високий), то треба вище підняти сидіння крісла, а під ноги підставити лавочку: чи що-небудь інше. Якщо стіл занадто низький, необхідно підкласти що-небудь під його ніжки.

Якщо при роботі часто приходиться дивитися на документи, треба установити підставку з оригіналом документа в одній площині і на одній, висоті з екраном. Якщо треба частіше дивитися на оригінал, чим на екран, необхідно повернути крісло й екран таким чином, щоб оригінал розміщався прямо перед очима, а комп'ютер ледве збоку.

Щогодини необхідно робити перерву в роботі. У цей час рекомендується робити вправи для зап'ясть. Нижче описаний можливий комплекс вправ.

- Покласти руку на край столу долонею вниз. Взявшись, за пальці, іншою рукою відвести кисть назад і утримувати протягом 5 секунд. Повторити для іншої руки.

- Злегка упертися рукою в стіл і на 5 секунд напружити пальці в зап'ястя. Повторити іншою рукою.

- Сильно зжати пальці , а потім розпрямити їх. Виконання приведених рекомендацій дозволяє скоротити вплив шкідливих, виробничих: факторів на організм людини.

Висновок

У розділі було розглянуто основні питання охорони праці під час експлуатації персонального комп'ютера в офісному приміщенні. Проведено аналіз небезпечних і шкідливих виробничих факторів, що можуть впливати на працівника, зокрема електромагнітного випромінювання, статичної електрики, недостатнього або надмірного освітлення, шуму, несприятливих параметрів мікроклімату, електробезпеки, зорового напруження та тривалого статичного навантаження. Визначено, що для забезпечення безпечних умов праці необхідно дотримуватися вимог щодо організації робочого місця, правильного розміщення монітора, клавіатури й робочого крісла, підтримання належного освітлення, температури, вологості та вентиляції приміщення. Запропоновані заходи, зокрема використання сучасних моніторів, антистатичних покриттів, раціонального режиму праці та відпочинку, а також виконання вправ для зменшення втоми, дають змогу знизити вплив шкідливих факторів на організм працівника. Отже, дотримання вимог охорони праці при роботі з комп'ютерною технікою є необхідною умовою збереження здоров'я користувачів, підвищення їхньої працездатності та забезпечення ефективної й безпечної експлуатації автоматизованої системи.

ВИСНОВКИ

В дипломній роботі було розроблено АРМ продавця-консультанта у структурі існуючої системи для магазину мобільного зв'язку, запропоновані різні методи для поліпшення ефективності роботи магазину, розроблена покращена система автоматизації робочого місця менеджера для більш стабільного і постійно зростаючого прибутку магазину.

Проєктування АРМ було проведено із застосуванням сучасних засобів проєктування інформаційних систем. Реалізоване програмне забезпечення мовою програмування Java із використанням засобів СУБД MySQL та серверу застосувань GlassFish.

У результаті виконання даного дипломного проєкту було створено Web-сайт з елементами функцій Internet-магазину для магазину мобільного зв'язку, що у подальшому дозволить створити повноцінну інтернет-крамницю на основі створеного АРМ.

Дана розробка дозволить спростити та суттєво прискорити процес представлення магазину мобільного зв'язку на ринку аналогічних послуг.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bloch, J. (2018). *Effective Java* (3rd ed.). Addison-Wesley Professional.
2. Gosling, J., Joy, B., Steele, G., Bracha, G., Buckley, A., Smith, D., & Bierman, G. (2023). *The Java language specification: Java SE 21 edition*. Oracle America, Inc.
3. Horstmann, C. S. (2024). *Core Java: Fundamentals* (13th ed.). Pearson.
4. JetBrains. (n.d.). *Build tools support in IntelliJ IDEA*. JetBrains Documentation.
5. JetBrains. (n.d.). *Code reference information in IntelliJ IDEA*. JetBrains Documentation.
6. JetBrains. (n.d.). *Debugging applications in IntelliJ IDEA*. JetBrains Documentation.
7. JetBrains. (n.d.). *Getting started with IntelliJ IDEA*. JetBrains Documentation.
8. JetBrains. (2026, April 29). *IntelliJ IDEA tutorials*. JetBrains Documentation.
9. JetBrains. (2026, May 19). *Support and assistance*. JetBrains Documentation.
10. JetBrains. (n.d.). *Version control integration in IntelliJ IDEA*. JetBrains Documentation.
11. Lindholm, T., Yellin, F., Bracha, G., Buckley, A., Smith, D., & Bierman, G. (2023). *The Java virtual machine specification: Java SE 21 edition*. Oracle America, Inc.
12. OpenJDK. (2023). *JDK 21 project: Features and release information*. OpenJDK Project.
13. OpenJDK. (2023). *JEP 440: Record patterns*. OpenJDK.
14. OpenJDK. (2023). *JEP 441: Pattern matching for switch*. OpenJDK.
15. OpenJDK. (2023). *JEP 444: Virtual threads*. OpenJDK.

16. Oracle. (n.d.). *Java Platform, Standard Edition 21 API specification*. Oracle Help Center.
17. Oracle. (n.d.). *Java SE 21 developer guides*. Oracle Help Center.
18. Oracle. (n.d.). *JDK 21 documentation*. Oracle Help Center.
19. Oracle. (2026, May 25). *MySQL Connector/J developer guide*. MySQL Documentation.
20. Oracle. (2026, June 9). *MySQL 8.4 reference manual*. MySQL Documentation.
21. Oracle. (n.d.). *MySQL 8.4 release notes*. MySQL Documentation.
22. Oracle. (n.d.). *MySQL Shell user guide*. MySQL Documentation.
23. Oracle. (n.d.). *MySQL Workbench manual*. MySQL Documentation.
24. Urma, R.-G., Fusco, M., & Mycroft, A. (2018). *Modern Java in action: Lambdas, streams, functional and reactive programming*. Manning Publications.

ДОДАТОК А – Технічне завдання

Призначення розробки системи

Назва програмного продукту: “Розробка АРМ менеджера інформаційної системи магазину побутової техніки”. Програмний продукт призначений автоматизації товаро—фінансових робіт підприємства.

Експлуатаційне призначення: при покупці товару, покупець повинен зареєструватися. Реєстрація проходить шляхом занесення особистих даних (ПІБ, телефон, адреса) в анкету з метою встановлення гарантійного строку для телефону, що купується. В даній системі автоматизуються наступні процеси: прийняти замовлення, оформити замовлення, закупка товару, продажу товару.

Мета розробки системи

Метою даної роботи є проектування такої інформаційної системи, яка б дозволила не тільки автоматизовано вести документацію і рух товарів, але й проводити обробку та аналіз замовлень з подальшим наданням звітності у різних розрізах.

Вимоги до структури і функціонування системи:

Програма повинна виконувати наступні дії:

- накопичувати інформацію про постачальників товару(назва, організація, адреса);
- зберігати інформації про клієнтів(ПІБ, адреса, код клієнта, телефон);
- зберігати інформацію про робітників(ПІБ, адреса, телефон, № паспорта, дата найма, заробітна плата);
- зберігати інформацію про товари(код,назва, № товару, кількість, марка, ціна);

- відсортувати параметри товарів по їхніх марках(код, назва марки);
- роздруковувати результат пошуку(назва марки, № товару, ціна, наявність товару)
- при наявності помилок користувача повідомляти його про цьому(помилка та заходи уникнення).

Вхідні дані вводяться працівником фірми із установленими правилами введення даних у програмі (тип, довжина, коректність).

Вимоги до надійності

Система повинна бути надійною і коректно реагувати на можливі нестандартні ситуації. Від її роботи залежить робота всього магазину, тому її надійність повинна бути дуже висока. Для того щоб оцінити надійність програми необхідно проаналізувати можливі аварійні ситуації:

- відключення електроживлення – для запобігання цієї ситуації встановлюється джерело безперебійного живлення, що дозволяє коректно закінчити роботу;
- вихід з ладу комп'ютера – для запобігання даної ситуації необхідно проводити систематичне тестування і перевірку працездатності;
- вихід з ладу локальної мережі – для запобігання цього необхідно періодично робити тестування функціонування мережі і не допускати повного виходу з ладу.

З розглянутого вище видно, що надійність системи навіть в аварійних ситуаціях залишається дуже велика.

Вимоги до безпеки

При роботі із системою, необхідно дотримувати правила техніки безпеки при роботі з ЕОМ. До роботи із системою допускаються тільки користувачі, що ознайомлені з посібником користувача.

Вимога до експлуатації, технічному обслуговуванню, ремонтіві і збереженню

До роботи допускаються тільки підготовлені користувачі, що пройшли інструктаж з техніки безпеки роботи з ЕОМ і ознайомлені з посібником користувача. При виникненні збоїв у роботі, користувач повинний сповістити про це адміністраторові системи.

Технічне обслуговування і ремонт устаткування здійснюється тільки кваліфікованими фахівцями.

Вимоги до захисту інформації від несанкціонованого доступу

Система повинна забезпечувати доступ тільки тим користувачам, що мають відповідні права доступу, а також вести схований облік усіх змін інформації.

Вимоги по схоронності інформації при аваріях

Повинна бути забезпечена збереження інформації при наступних аварійних ситуаціях:

- 1) відключення електроживлення;
- 2) вихід з ладу серверної станції;
- 3) вихід з ладу локальної комп'ютерної мережі.

Вимоги до захисту від зовнішніх впливів

Для безпечної роботи системи необхідно дотримувати правила техніки безпеки при роботі з ЕОМ і пожежної безпеки в робочому приміщенні.

Порядок контролю і приймання системи

Для контролю й прийому повинно бути надано керівництву оператори та опис програми, а також програма й методика випробування. Порядок контролю й прийому даної розробки здійснюється представником замовника в присутності представника розробника згідно з програмою та методикою випробувань. Якщо програма не пройшла випробування, виконавець зобов'язаний виправити помилки та недоліки в програмі, не більш ніж за 1 місяць із дня випробування.

За результатами прийому складається акт, який підписується представником замовника та представником розробника та утверджується керівниками організації-замовника й організації-розробника.

У випадку виявлення помилок під час прийому програмного виробу складається акт про виявлені помилки, який підписується представниками замовника та розробника й стверджується керівниками організації - замовника й організації - розробника.

Розробник повинен у продовж не більш як 1-го місяця виправити вказані зауваження, й повідомити замовника про повторне проведення перевірки, не пізніш як за 2 тижня до закінчення прийому програмного виробу.

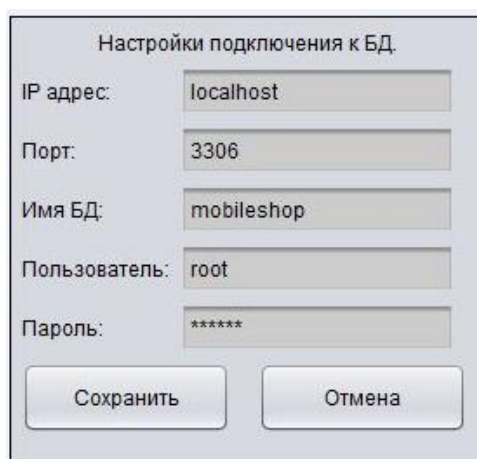
Вимоги до документації

До складу програмної документації повинні входити наступні документи:

- загальний опис системи;
- методика іспитів програмного забезпечення;
- посібник користувача.
- інструкція з формування і ведення бази даних.

ДОДАТОК Б – Інструкція користувача

Запускаємо програму на робочому столі, після цього потрібно ввести пароль для визначення користувача. Після цього натискаємо кнопку «Зберегти» і потрадаємо в головне вікно програми. Навігаційні кнопки та панель достатньо оснащені коментарями для розуміння та комфортної роботи з програмою.



Настройки подключения к БД.

IP адрес: localhost

Порт: 3306

Имя БД: mobileshop

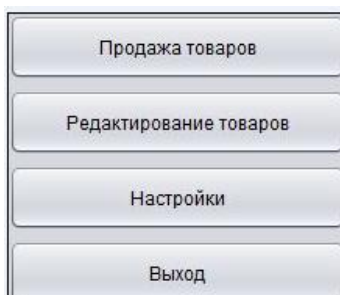
Пользователь: root

Пароль: *****

Сохранить Отмена

Рисунок 1 –Вікно налаштувань підключення до БД

На головному вікні зображені чотири кнопки: при натисканні на кнопку «Продажа товаров» відкривається вікно і ми можемо прослідкувати як продаються товари , с помощью кнопки «Редактирование товаров» при натисканні відкриється вікно редагування та додавання товарів і наступна кнопка «Настройки» при натисканні на цю кнопку відкриються настройки до БД.



Продажа товаров

Редактирование товаров

Настройки

Выход

Рисунок 2 – Головне вікно програми

В цьому вікні ми бачимо «Головне вікно продажу товарів».



це вікно пошуку товару по певним параметрам;



це таблиця допустимих до продажу товару;



це таблиця проданих товару;



оформити продаж товару;



Вихід.

№	Модель	Марка	Тип телеф...	Тип корпуса	Тип батареї	Цена	Количество
1	Nokia	5230	Смартфон	Моноблок	Li-Ion 1320...	1245	2
2	Samsung	E250	Мобильны...	Слайдер	1 Li-Ion 80...	870	1
3	Siemens	S32	КПК	Слайдер	1 Adveo 17...	2300	2
4	Sony Errics...	K800i	Смартфон	Моноблок	Li-Ion 1320...	1900	2

№	Параметр	Значение

Рисунок 3 –Головне вікно продажу товарів

ДОДАТОК В – Технічний опис

1 Призначення системи

Розроблена інформаційна система призначена для АРМ менеджера магазину мобільного зв'язку та формування звіту про продані товари, дозволяє редагувати та додавати дані і виконувати пошук по загальній базі даних товарів.

2 Умови застосування

Програма експлуатується на персональному комп'ютері (ПК) типа IBM PC/AT. Для роботи в діалоговому режимі використовується екран дисплея, клавіатура та маніпулятор типу «миша». Для підтримки графічного режиму необхідний адаптер SVGA Вхідні дані зберігаються на твердому диску комп'ютера. Програма працює під управлінням операційної системи Windows XP.

3 Опис задачі

Дана система покращує та спрощує роботу менеджера з продажу.

Задачі системи:

1. Додавання товарної позиції.
2. Редагування товарної позиції.
3. Внесення додаткової інформації про товар.
4. Внесення інформації про продаж товарів.
5. Пошук в загальній базі товарів.
6. Формування звіту про продані товари.

Виконання усіх перелічених задач реалізовано за допомогою окремих відповідних модулів розроблених за допомогою мови програмування Java та середовища NetBeans 7.3.

Запуск програми здійснюється при відкритті файлу Main.EXE.

Вхідні та вихідні дані:

Кожен з модулів має особисті вхідні та вихідні дані, які перелічено нижче:

Модуль “Настройки підключення до БД ”

Вхідні дані: IP адрес, ім'я БД, логін, пароль.

Вихідні дані: користувач отримує середовище для виконання роботи, завантажене за логіном та паролем.

Модуль “Продажа товарів”

Вхідні дані: вибір виду роботи з переліку

Вихідні дані: користувач отримує середовище для виконання обраної роботи.

Модуль „Редагування товару ”

Вхідні дані: при редагуванні товару ми має змогу змінити будь яку необхідну інформацію.

ДОДАТОК Г – SQL скрипт для розробленої БД

```
CREATE TABLE BatteryType
(idBatteryType  INTEGER NOT NULL,
 batteryType  VARCHAR(20) NOT NULL,
 PRIMARY KEY (idBatteryType));
```

```
CREATE TABLE BodyType
( idBodyType  INTEGER NOT NULL,
  bodyType  VARCHAR(20) NOT NULL,
  PRIMARY KEY (idBodyType));
```

```
CREATE TABLE Fillial
(idFillial  INTEGER NOT NULL,
 `name`  VARCHAR(40) NOT NULL,
 address  VARCHAR(100) NOT NULL,
 PRIMARY KEY (idFillial));
```

```
CREATE TABLE Mobilephone
(idMobilephone  INTEGER NOT NULL,
 price  INTEGER NOT NULL,
 count  INTEGER NOT NULL,
 idProduct  INTEGER NOT NULL,
 idFillial  INTEGER NOT NULL,
 PRIMARY KEY (idMobilephone));
```

```
CREATE TABLE MobilephoneParameters
(value  VARCHAR(100) NOT NULL,
 idMobilephone  INTEGER NOT NULL,
```

```
idParameters INTEGER NOT NULL,  
PRIMARY KEY (idMobilephone,idParameters));
```

```
CREATE TABLE Model  
(idModel INTEGER NOT NULL,  
model VARCHAR(35) NOT NULL,  
PRIMARY KEY (idModel));
```

```
CREATE TABLE `Order`  
(idOrder INTEGER NOT NULL,  
`count` INTEGER NOT NULL,  
orderDate DATE NOT NULL,  
price INTEGER NOT NULL,  
idMobilephone INTEGER NOT NULL,  
PRIMARY KEY (idOrder));
```

```
CREATE TABLE `Parameters`  
(idParameters INTEGER NOT NULL,  
`parameter` VARCHAR(45) NOT NULL,  
PRIMARY KEY (idParameters));
```

```
CREATE TABLE PhoneType  
(idPhoneType INTEGER NOT NULL,  
phoneType VARCHAR(20) NOT NULL,  
PRIMARY KEY (idPhoneType));
```

```
CREATE TABLE Product  
(mark VARCHAR(30) NOT NULL,  
idProduct INTEGER NOT NULL,  
idBatteryType INTEGER NOT NULL,
```



```

        idBodyType  INTEGER NOT NULL,
        idModel    INTEGER NOT NULL,
        idPhoneType INTEGER NOT NULL,
        PRIMARY KEY (idProduct));

ALTER TABLE Mobilephone
ADD FOREIGN KEY R_17 (idProduct) REFERENCES Product(idProduct);
ALTER TABLE Mobilephone
ADD FOREIGN KEY R_9 (idFillial) REFERENCES Fillial(idFillial);
ALTER TABLE MobilephoneParameters
ADD FOREIGN KEY R_19 (idMobilephone) REFERENCES
Mobilephone(idMobilephone);
ALTER TABLE MobilephoneParameters
ADD FOREIGN KEY R_20 (idParameters) REFERENCES
Parameters(idParameters);
ALTER TABLE `Order`
ADD FOREIGN KEY R_18 (idMobilephone) REFERENCES
Mobilephone(idMobilephone);
ALTER TABLE Product
ADD FOREIGN KEY R_13 (idPhoneType) REFERENCES
PhoneType(idPhoneType);
ALTER TABLE Product
ADD FOREIGN KEY R_14 (idBatteryType) REFERENCES
BatteryType(idBatteryType);
ALTER TABLE Product
ADD FOREIGN KEY R_15 (idBodyType) REFERENCES BodyType(idBodyType);
ALTER TABLE Product
ADD FOREIGN KEY R_16 (idModel) REFERENCES Model(idModel);

```

ДОДАТОК Д – Текст програми

```
package database;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;

//Интерфейс коннектора.
public interface Connector {
    public Connection getConnection();
    public ResultSet executeQuery(String sql) throws SQLException;
    public boolean execute(String sql) throws SQLException;
    public void executeUpdate(String sql) throws SQLException;
    public void reconnect();
}
```

Листинг 2 - файл < MySQLConnector>

```
package database;

import gui.MessageDialog;
import java.sql.*;
import java.util.Properties;
import settings.Settings;

//Подключение к БД.
public class MySQLConnector implements Connector {

    private static MySQLConnector instance;
    private Connection connection;

    private MySQLConnector() {
    }
    public static Connector getInstance() {
        if (instance == null) {
            instance = new MySQLConnector();
        }
        return instance;
    }
    @Override
    public void reconnect() {
        try {
            if (connection != null) {
                connection.close();
            }
        }
    }
}
```

```

        } catch (SQLException ex) {
            new MessageDialog(null,
true).showErrorMessage("Неудалось закрыть соединение");
        }
        connect();
    }
    @Override
    public Connection getConnection() {
        if (connection == null) {
            connect();
        }
        return connection;
    }
    //Получить настройки подключения.
    private Properties getProperties() {
        Settings settings = Settings.getInstance();
        return settings.getProperties();
    }

    //Подключение к БД. Возвращает true - если успешно, false -
если не удалось.
    private boolean connect() {
        try {
            Properties properties = getProperties();
            String ip = properties.getProperty("ip");
            String port = properties.getProperty("port");
            String dbname = properties.getProperty("dbname");
            String user = properties.getProperty("user");
            String password = properties.getProperty("password");
            String url = "jdbc:mysql://" + ip + ":" + port + "/" +
dbname;
            connection = DriverManager.getConnection(url, user,
password);
            return true;
        } catch (SQLException ex) {
            return false;
        }
    }
    //SQL запрос с возвратом значений.
    @Override
    public ResultSet executeQuery(String sql) throws SQLException
{
        ResultSet resultSet = null;
        if (connect()) {

```

```

        Statement query =
connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);
        resultSet = query.executeQuery(sql);
    }
    return resultSet;
}

//SQL запрос с возвратом true - если успешно, иначе false.
@Override
public boolean execute(String sql) throws SQLException {
    boolean executed = false;
    if (connect()) {
        Statement query = connection.createStatement();
        executed = query.execute(sql);
    }
    return executed;
}
@Override
public void executeUpdate(String sql) throws SQLException {
    if (connect()) {
        Statement query = connection.createStatement();
        query.executeUpdate(sql);
    }
}
@Override
protected void finalize() throws Throwable {
    super.finalize();
    if (connection != null) {
        connection.close();
    }
}
}

```

Листинг 3 - файл < Executor>

```

package database;

import gui.MessageDialog;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
//Класс отвечающий за запросы к БД.
public class Executor {
    Connector connector;

```

```

public Executor() {
    connector = MySQLConnector.getInstance();
}
//Поиск всех телефонов из БД. all = true показывает и нелефоны
которых нет в наличии
public ResultSet findAllMobilephones(boolean all) {
    ResultSet resultSet = null;
    try {
        if (all) {
            resultSet =
connector.executeQuery(Queries.SELECT_ALL_FROM_MOBILEPHONES + " ORDER
BY `idMobilephone`");
        } else {
            resultSet =
connector.executeQuery(Queries.SELECT_ALL_FROM_MOBILEPHONES + " AND
m.`count` > 0 ORDER BY `idMobilephone`");
        }
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось найти все телефоны.");
    }
    return resultSet;
}
//Поиск одного телефона по его ИД
public ResultSet findMobilephoneById(int idMobilephone) {
    ResultSet resultSet = null;
    Connection connection = connector.getConnection();
    try {
        PreparedStatement statement =
connection.prepareStatement(Queries.SELECT_MOBILEPHONE_BY_ID);
        statement.setInt(1, idMobilephone);
        resultSet = statement.executeQuery();
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось найти телефон.");
    }
    return resultSet;
}
//Параметры определенного телефона
public ResultSet mobileParametersById(int idMobilephone) {
    ResultSet resultSet = null;
    Connection connection = connector.getConnection();
    try {

```

```

        PreparedStatement statement =
connection.prepareStatement(Queries.SELECT_FROM_PARAMETER_BY_ID);
        statement.setInt(1, idMobilephone);
        resultSet = statement.executeQuery();
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось найти параметры телефона.");
    }
    return resultSet;
}

//Все проданные телефоны. Таблица ОРдерс
public ResultSet findAllSelledPhones() {
    ResultSet resultSet = null;
    try {
        resultSet =
connector.executeQuery(Queries.SELECT_FROM_SELLED_MOBILEPHONES);
    } catch (SQLException ex) {
        new MessageDialog(null, true).showErrorMessage("Список
продаж пуст.");
    }
    return resultSet;
}

//Продажа телефона.
public void buyMobilephone(int id, int idMobilephone, int
count, int price, int max) {
    try {
        Connection connection = connector.getConnection();
        PreparedStatement statement =
connection.prepareStatement(Queries.INSERT_INTO_ORDERS);
        statement.setInt(1, id);
        statement.setInt(2, count);
        statement.setDate(3, new
java.sql.Date(System.currentTimeMillis()));
        statement.setInt(4, price);
        statement.setInt(5, idMobilephone);
        statement.executeUpdate();
        takeMobilePhone(idMobilephone, count, max);
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось заполнить таблицу покупок.");
    }
}

```

```

//Вычитание телефона из БД
public void takeMobilePhone(int idMobilephone, int count, int
max) {
    try {
        Connection connection = connector.getConnection();
        PreparedStatement statement =
connection.prepareStatement(Queries.SELL_MOBILEPHONE);
        statement.setInt(1, max - count);
        statement.setInt(2, idMobilephone);
        statement.executeUpdate();
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось изменить таблицу телефонов.");
    }
}
//Получить всободный индекс таблицы.
public int getFreeId(String SQL) {
    int id = 1;
    try {
        ResultSet rs = connector.executeQuery(SQL);
        while (rs.next()) {
            if (rs.getInt(1) == id) {
                id++;
            }
        }
        return id;
    } catch (SQLException ex) {
        return 0;
    }
}
//Получить любую таблицу справочник.
public ResultSet getAllFromTable(String SQL) {
    ResultSet resultSet = null;
    try {
        resultSet = connector.executeQuery(SQL);
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось получить мануал.");
    }
    return resultSet;
}
//Получить все модели телефонов.
public ResultSet getAllModel() {
    return getAllFromTable(Queries.SELECT_ALL_FROM_MODEL);
}

```

```

    }
    //Получить все типы телефонов.
    public ResultSet getAllPhoneType() {
        return getAllFromTable(Queries.SELECT_ALL_FROM_PHONETYPE);
    }
    //Получить все типы батарей.
    public ResultSet getAllBatteryType() {
        return
getAllFromTable(Queries.SELECT_ALL_FROM_BATTERYTYPE);
    }
    //Получить все типы корпуса.
    public ResultSet getAllBodyType() {
        return getAllFromTable(Queries.SELECT_ALL_FROM_BODYTYPE);
    }

    //Получить все филиалы.
    public ResultSet getAllFillials() {
        return getAllFromTable(Queries.SELECT_ALL_FROM_FILLIALS);
    }
    //Получить все мобильные телефоны с подолнительным СКУЛ кодом.
    public ResultSet findAllMobilephones(StringBuilder SQL) {
        ResultSet resultSet = null;
        try {
            resultSet =
connector.executeQuery(Queries.SELECT_ALL_FROM_MOBILEPHONES + SQL);
        } catch (SQLException ex) {
            new MessageDialog(null,
true).showErrorMessage("Неудалось получить весь список телефонов.");
        }
        return resultSet;
    }
    //Добавить запись в справочник.
    public void insertManuals(String addSQL, int index, String
item) {
        try {
            Connection connection = connector.getConnection();
            PreparedStatement statement =
connection.prepareStatement(addSQL);
            statement.setInt(1, index);
            statement.setString(2, item);
            statement.executeUpdate();
        } catch (SQLException ex) {
            new MessageDialog(null,
true).showErrorMessage("Неудалось добавить '" + item + "'");

```



```

        }
    }
    //Обновить запись в справочнике.
    public void updateManuals(String addSQL, int index, String
item) {
        try {
            Connection connection = connector.getConnection();
            PreparedStatement statement =
connection.prepareStatement(addSQL);
            statement.setString(1, item);
            statement.setInt(2, index);
            statement.executeUpdate();
        } catch (SQLException ex) {
            new MessageDialog(null,
true).showErrorMessage("Неудалось добавить '" + item + "'");
        }
    }
    //Добавить продукт.
    public int addProduct(String mark, int batteryID, int bodyID,
int modelID, int phoneID) {
        int id = -1;
        try {
            Connection connection = connector.getConnection();
            PreparedStatement statement =
connection.prepareStatement(Queryes.INSERT_INTO_PRODUCT);
            statement.setString(1, mark);
            id = getFreeId(Queryes.INDEX_FROM_PRODUCT);
            statement.setInt(2, id);
            statement.setInt(3, batteryID);
            statement.setInt(4, bodyID);
            statement.setInt(5, modelID);
            statement.setInt(6, phoneID);
            statement.executeUpdate();
        } catch (SQLException ex) {
            new MessageDialog(null,
true).showErrorMessage("Неудалось добавить новый телефон.");
        }
        return id;
    }
    //Добавить мобильный телефон.
    public void addMobilephone(int idMobilephone, int price, int
count, int idProduct, int idFillial) {
        try {
            Connection connection = connector.getConnection();

```

```

        PreparedStatement statement =
connection.prepareStatement(Queries.INSERT_INTO_MOBILEPHONES);
        statement.setInt(1, idMobilephone);
        statement.setInt(2, price);
        statement.setInt(3, count);
        statement.setInt(4, idProduct);
        statement.setInt(5, idFillial);
        statement.executeUpdate();
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось добавить новый телефон.");
    }
}

//обновить продукт.
public void updateProduct(int idProduct, String mark, int
batteryID, int bodyID, int modelID, int phoneID) {
    try {
        Connection connection = connector.getConnection();
        PreparedStatement statement =
connection.prepareStatement(Queries.UPDATE_PRODUCT);
        statement.setString(1, mark);
        statement.setInt(2, batteryID);
        statement.setInt(3, bodyID);
        statement.setInt(4, modelID);
        statement.setInt(5, phoneID);
        statement.setInt(6, idProduct);
        statement.executeUpdate();
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось изменить телефон.");
    }
}

//Обновить мобильный телефон.
public void updateMobilephone(int idMobilephone, int price,
int count, int idFillial) {
    try {
        Connection connection = connector.getConnection();
        PreparedStatement statement =
connection.prepareStatement(Queries.UPDATE_MOBILEPHONES);
        statement.setInt(1, price);
        statement.setInt(2, count);
        statement.setInt(3, idFillial);
        statement.setInt(4, idMobilephone);
        statement.executeUpdate();
    }
}

```

```

        } catch (SQLException ex) {
            new MessageDialog(null,
true).showErrorMessage("Неудалось изменить телефон.");
        }
    }
    //Удалить мобильный телефон по ИД
    public void removeMobilephone(Object id) {
        try {
            Connection connection = connector.getConnection();
            int idMobilephone = Integer.valueOf(id.toString());
            PreparedStatement statement =
connection.prepareStatement(Queries.SELECT_PRODUCT_ID);
            statement.setInt(1, idMobilephone);
            ResultSet rs = statement.executeQuery();
            int idProduct = 0;
            if (rs.next()) {
                idProduct = rs.getInt(1);
            }
            statement =
connection.prepareStatement(Queries.DELETE_PARAMETERS_BY_ID);
            statement.setInt(1, idMobilephone);
            statement.execute();
            statement =
connection.prepareStatement(Queries.DELETE_MOBILEPHONE_BY_ID);
            statement.setInt(1, idMobilephone);
            statement.executeUpdate();
            try {
                removeProductByIdMobilephone(connection,
idProduct);
            } catch (SQLException ex) {
            }
        } catch (SQLException ex) {
            new MessageDialog(null,
true).showErrorMessage("Неудалось удалить телефон.");
        }
    }
    //Удалить продукт по ИД мобильного телефона.
    private void removeProductByIdMobilephone(int idMobilephone,
Connection connection) throws SQLException {
        PreparedStatement statement;
        statement =
connection.prepareStatement(Queries.SELECT_PRODUCT_ID);
        statement.setInt(1, idMobilephone);
        ResultSet rs = statement.executeQuery();

```

```

        int idProduct = 0;
        if (rs.next()) {
            idProduct = rs.getInt(1);
        }
        statement =
connection.prepareStatement(Queries.DELETE_PRODUCT_BY_ID);
        statement.setInt(1, idProduct);
        statement.executeUpdate();
    }
    //Удалить продукт по ИД.
    private void removeProductByIdMobilephone(Connection
connection, int idProduct) throws SQLException {
        PreparedStatement statement =
connection.prepareStatement(Queries.DELETE_PRODUCT_BY_ID);
        statement.setInt(1, idProduct);
        statement.executeUpdate();
    }
    //Удалить параметр мобильного телефона
    public void removeMobilephoneParameter(int idMobilephone, int
idParameter) {
        try {
            Connection connection = connector.getConnection();
            PreparedStatement statement =
connection.prepareStatement(Queries.DELETE_PARAMETER_BY_IDM_IDP);
            statement.setInt(1, idMobilephone);
            statement.setInt(2, idParameter);
            statement.execute();
        } catch (SQLException ex) {
            new MessageDialog(null,
true).showErrorMessage("Неудалось удалить параметр телефона.");
        }
    }
    //Добавить параметр.
    public void insertParameter(int index, String param) {
        try {
            Connection connection = connector.getConnection();
            PreparedStatement statement =
connection.prepareStatement(Queries.INSERT_INTO_PARAMETERS);
            statement.setInt(1, index);
            statement.setString(2, param);
            statement.execute();
        } catch (SQLException ex) {
            new MessageDialog(null,
true).showErrorMessage("Неудалось добавить параметр.");

```

```

    }
}
//Добавить параметр мобильного телефона
public void insertMobilephoneParameter(int idMobilephone, int
index, String text) {
    try {
        Connection connection = connector.getConnection();
        PreparedStatement statement =
connection.prepareStatement(Queryes.INSERT_INTO_MOBILEPHONE_PARAMETERS
);
        statement.setString(1, text);
        statement.setInt(2, idMobilephone);
        statement.setInt(3, index);
        statement.execute();
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось добавить параметр телефона.");
    }
}
//Обновить параметр мобильного телефона.
public void updateMobilephoneParameter(int idMobilephone, int
index, String text) {
    try {
        Connection connection = connector.getConnection();
        PreparedStatement statement =
connection.prepareStatement(Queryes.UPDATE_MOBILEPHONE_PARAMETER);
        statement.setString(1, text);
        statement.setInt(2, idMobilephone);
        statement.setInt(3, index);
        statement.executeUpdate();
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось добавить параметр телефона.");
    }
}
//Удалить продажу.
public void removeOrder(int idOrder) {
    Connection connection = connector.getConnection();
    try {
        PreparedStatement statement =
connection.prepareStatement(Queryes.DELETE_ORDER_BY_ID);
        statement.setInt(1, idOrder);
        statement.execute();
    } try {

```

```

        removeProductByIdMobilephone(idOrder, connection);
    } catch (SQLException ex) {
    }
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось удалить продажу.");
    }
}
//Получить параметр по ИД
public String getParamNameByID(int idParameter) {
    try {
        Connection connection = connector.getConnection();
        PreparedStatement statement =
connection.prepareStatement(Queryes.SELECT_PARAMETER_NAME_BY_ID);
        statement.setInt(1, idParameter);
        ResultSet rs = statement.executeQuery();
        if (rs.next()) {
            return rs.getString(1);
        }
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось найти ID телефона.");
    }
    return "";
}
//Удалить запись из справочника.
public void removeManual(String sql, Integer index) {
    try {
        Connection connection = connector.getConnection();
        PreparedStatement statement =
connection.prepareStatement(sql);
        statement.setInt(1, index);
        statement.executeUpdate();
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось найти ID телефона.");
    }
}

public void insertFillial(int index, String name, String
address) {
    try {
        Connection connection = connector.getConnection();

```

```

        PreparedStatement statement =
connection.prepareStatement(Queryes.INSERT_INTO_FILLIALS);
        statement.setInt(1, index);
        statement.setString(2, name);
        statement.setString(3, address);
        statement.execute();
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось добавить филиал.");
    }
}

    public void updateFillial(int index, String name, String
address) {
        try {
            Connection connection = connector.getConnection();
            PreparedStatement statement =
connection.prepareStatement(Queryes.UPDATE_FILLIAL);
            statement.setString(1, name);
            statement.setString(2, address);
            statement.setInt(3, index);
            statement.executeUpdate();
        } catch (SQLException ex) {
            new MessageDialog(null,
true).showErrorMessage("Неудалось изменить филиал.");
        }
    }

    public void removeFillial(int index) {
        try {
            Connection connection = connector.getConnection();
            PreparedStatement statement =
connection.prepareStatement(Queryes.DELETE_FILLIAL_BY_ID);
            statement.setInt(1, index);
            statement.executeUpdate();
        } catch (SQLException ex) {
            new MessageDialog(null,
true).showErrorMessage("Неудалось удалить филиал.");
        }
    }
}

Листинг 4 - файл < Queryes>
package database;

public interface Queryes {

```

```

        public static String SELECT_ALL_FROM_MOBILEPHONES = "SELECT
m.`idMobilephone` AS '№', mo.model AS 'Модель',"
        + " p.mark AS 'Марка', pt.`phoneType` AS 'Тип
телефона', bdt.`bodyType` AS 'Тип корпуса',"
        + " bt.`batteryType` AS 'Тип батареи', m.price AS
'Цена', m.`count` AS 'Количество', f.`name` AS 'Отделение'"
        + " FROM mobilephone m, product p, batterytype bt,
model mo, phonetype pt, bodytype bdt, fillial f "
        + "WHERE m.`idProduct` = p.`idProduct` AND
p.`idPhoneType` = pt.`idPhoneType` AND p.`idModel` = mo.`idModel`"
        + " AND p.`idBodyType` = bdt.`idBodyType` AND
p.`idBatteryType` = bt.`idBatteryType` "
        + "AND m.`idFillial` = f.`idFillial`";

        public static String SELECT_FROM_MOBILEPHONES_BY_MODEL =
"SELECT m.`idMobilephone` AS '№', mo.model AS 'Модель',"
        + " p.mark AS 'Марка', pt.`phoneType` AS 'Тип
телефона', bdt.`bodyType` AS 'Тип корпуса', bt.`batteryType` "
        + "AS 'Тип батареи', m.price AS 'Цена', m.`count` AS
'Количество', f.`name` AS 'Отделение' "
        + "FROM mobilephone m, product p, batterytype bt,
model mo, phonetype pt, bodytype bdt, fillial f "
        + "WHERE m.`idProduct` = p.`idProduct` AND
p.`idPhoneType` = pt.`idPhoneType` AND p.`idModel` = mo.`idModel`"
        + " AND mo.model = ? AND p.`idBodyType` =
bdt.`idBodyType` AND p.`idBatteryType` = bt.`idBatteryType` "
        + "AND m.`idFillial` = f.`idFillial`";

        public static String SELECT_FROM_SELLED_MOBILEPHONES = "SELECT
o.`idOrder` AS '№ заказа', m.`idMobilephone` "
        + "AS '№ телефона', mo.model AS 'Модель', p.mark AS
'Марка', o.price AS 'Цена', o.`count` "
        + "AS 'Количество', f.`name` AS 'Отделение',
o.`orderDate` AS 'Дата' FROM `order` o, mobilephone m,"
        + " product p, model mo, fillial f WHERE
o.`idMobilephone` = m.`idMobilephone`, m.`idFillial` = f.`idFillial`"
        + " AND m.`idProduct` = p.`idProduct` AND p.`idModel`
= mo.`idModel` ORDER BY o.`idOrder`";

        public static String SELECT_FROM_PARAMETER_BY_ID = "SELECT
mp.`idParameters` AS '№', p.`parameter` AS 'Параметр', mp.`value` AS
'Значение' "
        + "FROM mobilephoneparameters mp, `parameters` p WHERE
mp.`idMobilephone` = ? "
        + "AND mp.`idParameters` = p.`idParameters` ORDER BY
mp.`idParameters`";

```



```

        public static String SELECT_MOBILEPHONE_BY_ID = "SELECT
p.`idModel`, p.mark, p.`idBodyType`, p.`idPhoneType`,
p.`idBatteryType`, m.price, m.`count`, m.`idProduct`, m.`idFillial` "
        + "FROM mobilephone m, product p WHERE
m.`idMobilephone` = ? AND p.`idProduct` = m.`idProduct`";
        public static String SELECT_ALL_FROM_MODEL = "select
m.`idModel` AS '№', m.model AS 'Модель' from model m";
        public static String SELECT_ALL_FROM_BODYTYPE = "select
bt.`idBodyType` AS '№', bt.`bodyType` AS 'Тип корпуса' from bodytype
bt";
        public static String SELECT_ALL_FROM_BATTERYTYPE = "select
b.`idBatteryType` AS '№', b.`batteryType` AS 'Тип батареи' from
batterytype b";
        public static String SELECT_ALL_FROM_PHONETYPE = "select
pt.`idPhoneType` AS '№', pt.`phoneType` AS 'Тип телефона' from
phonetype pt";
        public static String SELECT_ALL_FROM_PARAMETERS_NAME = "select
p.`idParameters` AS '№', p.`parameter` AS 'Параметр' from `parameters`
p";
        public static String SELECT_ALL_FROM_FILLIALS = "SELECT
f.`idFillial` AS '№', f.`name` AS 'Название', f.address AS 'Адрес'
FROM fillial f";
        public static String SELECT_PRODUCT_ID = "SELECT `idProduct`
FROM mobilephone WHERE `idMobilephone` = ?";
        public static String SELECT_PARAMETER_NAME_BY_ID = "SELECT
p.`parameter` FROM `parameters` p WHERE p.`idParameters` = ?";

        public static String SELL_MOBILEPHONE = "UPDATE mobilephone m
SET m.`count` = ? WHERE m.`idMobilephone`=?"

        public static String INSERT_INTO_ORDERS = "INSERT INTO `order`
(`idOrder`, `count`, `orderDate`, price, `idMobilephone`) VALUES(?, ?, ?,
?, ?)";
        public static String INSERT_INTO_MODEL = "INSERT INTO model
(`idModel`, model) VALUES (?, ?)";
        public static String INSERT_INTO_BODYTYPE = "INSERT INTO
bodytype (`idBodyType`, `bodyType`) VALUES (?, ?)";
        public static String INSERT_INTO_BATTERYTYPE = "INSERT INTO
batterytype (`idBatteryType`, `batteryType`) VALUES (?, ?)";
        public static String INSERT_INTO_PHONETYPE = "INSERT INTO
phonetype (`idPhoneType`, `phoneType`) VALUES (?, ?)";
        public static String INSERT_INTO_FILLIALS = "INSERT INTO
fillial(`idFillial`, `name`, address) VALUES(?, ?, ?)";

```

```

        public static String INSERT_INT0_PRODUCT = "INSERT INTO
product (mark, `idProduct`,
`idBatteryType`, `idBodyType`, `idModel`, `idPhoneType`) VALUES
(?,?,?, ?, ?, ?)";
        public static String INSERT_INT0_MOBILEPHONES = "INSERT INTO
mobilephone(`idMobilephone`, price, `count`, `idProduct`,
`idFillial`)VALUES(?,?,?,?,?)";
        public static String INSERT_INT0_PARAMETERS = "INSERT INTO
`parameters` (`idParameters`, `parameter`) VALUES (?, ?)";
        public static String INSERT_INT0_MOBILEPHONE_PARAMETERS =
"INSERT INTO mobilephoneparameters (`value`, `idMobilephone`,
`idParameters`) VALUES (?,?,?)";

        public static String INDEX_FROM_PRODUCT = "SELECT `idProduct`
FROM product";
        public static String INDEX_INDEX_MOBILEPHONES = "SELECT
`idMobilephone` FROM mobilephone";

        public static String DELETE_PHONE_PARAMETERS = "DELETE FROM
mobilephoneparameters WHERE mobilephoneparameters.`idMobilephone` =
?";
        public static String DELETE_MOBILEPHONE = "DELETE FROM
mobilephone WHERE mobilephone.`idMobilephone` = ?";
        public static String DELETE_PARAMETERS_BY_ID = "DELETE FROM
mobilephoneparameters WHERE `idMobilephone` = ?";
        public static String DELETE_MOBILEPHONE_BY_ID = "DELETE FROM
mobilephone WHERE `idMobilephone` = ?";
        public static String DELETE_PRODUCT_BY_ID = "DELETE FROM
product WHERE `idProduct` = ?";
        public static String DELETE_PARAMETER_BY_IDM_IDP = "DELETE
FROM mobilephoneparameters WHERE `idMobilephone` = ? AND
`idParameters` = ?";
        public static String DELETE_ORDER_BY_ID = "DELETE FROM `order`
WHERE `idOrder` = ?";
        public static String DELETE_BODYTYPE_BY_ID = "DELETE FROM
bodytype WHERE `idBodyType` = ?";
        public static String DELETE_PHONETYPE_BY_ID = "DELETE FROM
phonetype WHERE `idPhoneType` = ?";
        public static String DELETE_BATTERYTYPE_BY_ID = "DELETE FROM
batterytype WHERE `idBatteryType` = ?";
        public static String DELETE_MODEL_BY_ID = "DELETE FROM model
WHERE `idModel` = ?";
        public static String DELETE_PARAMETER_BY_ID = "DELETE FROM
`parameters` WHERE `idParameters` = ?";

```

```

        public static String DELETE_FILLIAL_BY_ID = "DELETE FROM
fillial WHERE fillial.`idFillial` = ?";

        public static String UPDATE_MOBILEPHONE_PARAMETER = "UPDATE
mobilephoneparameters SET `value` = ? WHERE `idMobilephone` = ? AND
`idParameters` = ?";
        public static String UPDATE_PRODUCT = "UPDATE product SET mark
= ?, `idBatteryType` = ?, `idBodyType` = ?, `idModel` = ?,
`idPhoneType` = ? "
            + "WHERE `idProduct` = ?";
        public static String UPDATE_MOBILEPHONES = "UPDATE mobilephone
SET price = ?, `count` = ?, `idFillial` = ? WHERE `idMobilephone` =
?";
        public static String UPDATE_BODYTYPE = "UPDATE bodytype SET
`bodyType` = ? WHERE `idBodyType` = ?";
        public static String UPDATE_PHONETYPE = "UPDATE phonetype SET
`phoneType` = ? WHERE `idPhoneType` = ?";
        public static String UPDATE_BATTERYTYPE = "UPDATE batterytype
SET `batteryType` = ? WHERE `idBatteryType` = ?";
        public static String UPDATE_MODEL = "UPDATE model SET model =
? WHERE `idModel` = ?";
        public static String UPDATE_PARAMETERS = "UPDATE `parameters`
SET `parameter` = ? WHERE `idParameters` = ?";
        public static String UPDATE_FILLIAL = "UPDATE fillial SET
`name` = ? , address = ? WHERE `idFillial` = ?";
    }

```

Пакет <guilogic>

Листинг 1 - файл <AddFillialLogic>

```
package guilogic;
```

```
import gui.AddFillialDialog;
```

```
public class AddFillialLogic {
```

```
    private AddFillialDialog dialog;
    private boolean confirm;
```

```
    public AddFillialLogic() {
        dialog = new AddFillialDialog(null, true, this);
    }
```

```
    public void showAddFillialDialog() {
        dialog.getjTextFieldName().setText("");
        dialog.getjTextAreaAddress().setText("");
    }
```

```

        confirm = false;
        dialog.setVisible(true);
    }
    public void showEditFillialDialog(String name, String address)
{
        dialog.getjTextFieldName().setText(name);
        dialog.getjTextAreaAddress().setText(address);
        confirm = false;
        dialog.setVisible(true);
    }
    public String getName() {
        return dialog.getjTextFieldName().getText();
    }
    public String getAddress() {
        return dialog.getjTextAreaAddress().getText();
    }
    public boolean check() {
        if ("".equals(dialog.getjTextFieldName().getText())) {
            return false;
        }
        if ("".equals(dialog.getjTextAreaAddress().getText())) {
            return false;
        }
        return true;
    }
    public boolean isConfirmed() {
        return confirm;
    }
    public void confirm() {
        confirm = true;
        dialog.setVisible(false);
    }
    public void cancel() {
        confirm = false;
        dialog.setVisible(false);
    }
}

```

Листинг 2 - файл <BuyPhoneLogic>

```
package guilogic;
```

```
import database.Executor;
import gui.BuyPhoneDialog;
```

```

public class BuyPhoneLogic {

    private BuyPhoneDialog dialog;
    private Executor executor = new Executor();
    private int price = 0, max, value = 1, id;
    private boolean isSelled = false;

    public BuyPhoneLogic() {
        dialog = new BuyPhoneDialog(null, true, this);
    }

    public boolean isSelled() {
        return isSelled;
    }

    public void showDialog(int id, String mark, String model, int
price, int maximum) {
        this.price = price;
        this.id = id;
        this.max = maximum;
        dialog.initSpinner(maximum);
        dialog.setTextInLabel1("Марка: " + mark);
        dialog.setTextInLabel5("Модель: " + model);
        dialog.setTextInLabel7("Цена: " + price);
        isSelled = false;
        dialog.setVisible(true);
    }

    public void calculatePrice() {
        value = (int) dialog.getSpinner().getValue();
        dialog.setTextInLabel7("Цена: " + (price * value));
    }

    public void exitButton() {
        dialog.setVisible(false);
    }

    public void confirmButton() {
        int orderId = executor.getFreeId("SELECT * FROM `order`");
        executor.buyMobilephone(orderId, id, value, value * price,
max);

        isSelled = true;
        dialog.setVisible(false);
    }
}

```

Листинг 3 - файл <FindPhoneLogic>

```

package guilogic;

```

```

import database.Executor;
import gui.FindPhoneDialog;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.DefaultComboBoxModel;

public class FindPhoneLogic {

    private FindPhoneDialog dialog;
    private final MobilephonesDialogLogic logic;

    public FindPhoneLogic(MobilephonesDialogLogic logic) {
        dialog = new FindPhoneDialog(null, true, this);
        this.logic = logic;
    }

    public void show() {
        Executor executor = new Executor();

        dialog.setjComboBox1(createComboBoxModel(executor.getAllModel()));

        dialog.setjComboBox2(createComboBoxModel(executor.getAllPhoneType()));

        dialog.setjComboBox3(createComboBoxModel(executor.getAllBodyType()));

        dialog.setjComboBoxFillial(createComboBoxModel(executor.getAllFillials()));

        dialog.setAllCheckBoxUnselected();
        dialog.clearJTextFields();
        dialog.setVisible(true);
    }

    public void confirmButton() {
        logic.updateTable(getPhonesByParams());
        dialog.setVisible(false);
    }

    private ResultSet getPhonesByParams() {
        StringBuilder SQL = new StringBuilder(" AND m.`count` > 0
");
        if (dialog.getjCheckBox4()) {
            int fillialId = new
Scanner(dialog.getjComboBoxFillial().getSelectedItem().toString()).nextInt();

```

```

        SQL.append("AND
m.`idFillial`=").append(fillialId).append(" ");
    }
    if (dialog.getjCheckBox1()) {
        int modelId = new
Scanner(dialog.getjComboBox1().getSelectedItem().toString()).nextInt()
;
        SQL.append("AND
mo.`idModel`=").append(modelId).append(" ");
    }
    if (dialog.getjCheckBox2()) {
        int phoneTypeId = new
Scanner(dialog.getjComboBox2().getSelectedItem().toString()).nextInt()
;
        SQL.append("AND
pt.`idPhoneType`=").append(phoneTypeId).append(" ");
    }
    if (dialog.getjCheckBox3()) {
        int bodyTypeId = new
Scanner(dialog.getjComboBox3().getSelectedItem().toString()).nextInt()
;
        SQL.append("AND
bdt.`idBodyType`=").append(bodyTypeId).append(" ");
    }
    if (dialog.getjCheckBox5()) {
        int start = dialog.getTextField1();
        int end = dialog.getTextField2();
        if (start > end || end == 0) {
            end = Integer.MAX_VALUE;
        }
        SQL.append("AND m.price >=").append(start).append("
AND m.price <=").append(end).append(" ");
    }
    return new Executor().findAllMobilephones(SQL);
}

private DefaultComboBoxModel<String>
createComboBoxModel(ResultSet rs) {
    DefaultComboBoxModel<String> boxModel = new
DefaultComboBoxModel();
    try {
        while (rs.next()) {
            boxModel.addElement(rs.getInt(1) + " " +
rs.getString(2));
        }
    }
}

```

```

        } catch (SQLException ex) {

Logger.getLogger(FindPhoneLogic.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return boxModel;
    }
}

```

Листинг 4 - файл <SettingsDialogLogic>

```

package guilogic;

import com.sun.xml.internal.fastinfoset.util.CharArray;
import database.MySQLConnector;
import gui.MessageDialog;
import gui.SettingsDialog;
import java.awt.Frame;
import java.util.Properties;
import settings.Settings;

public class SettingsDialogLogic {

    private Settings settings;
    private Properties properties;
    private SettingsDialog settingsDialog;

    public SettingsDialogLogic(Frame parent) {
        settingsDialog = new SettingsDialog(parent, true, this);
    }

    public void showPropertiesDialog() {
        initTextFields();
        settingsDialog.setVisible(true);
    }

    private void initTextFields() {
        settings = Settings.getInstance();
        properties = settings.getProperties();

settingsDialog.setTextInField1(properties.getProperty("ip"));

settingsDialog.setTextInField2(properties.getProperty("port"));

settingsDialog.setTextInField4(properties.getProperty("dbname"));

settingsDialog.setTextInField3(properties.getProperty("user"));

```



```

settingsDialog.setPassword(properties.getProperty("password"));
    }
    private boolean checkFields() {
        if ("".equals(settingsDialog.getTextFromField1())) {
            return false;
        }
        if ("".equals(settingsDialog.getTextFromField2())) {
            return false;
        }
        if ("".equals(settingsDialog.getTextFromField4())) {
            return false;
        }
        if ("".equals(settingsDialog.getTextFromField3())) {
            return false;
        }
        return true;
    }
    public void confirm() {
        if (checkFields()) {
            properties.setProperty("ip",
settingsDialog.getTextFromField1());
            properties.setProperty("port",
settingsDialog.getTextFromField2());
            properties.setProperty("dbname",
settingsDialog.getTextFromField4());
            properties.setProperty("user",
settingsDialog.getTextFromField3());
            char[] password = settingsDialog.getPassword();
            StringBuilder sb = new StringBuilder(new
CharArray(password, 0, password.length, false));
            properties.setProperty("password", sb.toString());
            settings.saveProperties();
            MySQLConnector.getInstance().reConnect();
            settingsDialog.setVisible(false);
        } else {
            new MessageDialog(null, true).showErrorMessage("Поля
не должны быть пустыми.");
        }
    }

    public void cancel() {
        settingsDialog.setVisible(false);
    }

```

```
}
```

Листинг 5 - файл <AddManualLogic>

```
package guilogic;

import gui.AddManualDialog;

public class AddManualLogic {

    private AddManualDialog addManualDialog;
    private String aValue;
    private boolean confirm;

    AddManualLogic() {
        this.addManualDialog = new AddManualDialog(null, true,
this);
    }
    public void showAddDialog(String name) {
        addManualDialog.getjLabel2().setText(name);
        aValue = "";
        addManualDialog.getjTextField1().setText(aValue);
        confirm = false;
        addManualDialog.setVisible(true);
    }
    public void showEditDialog(String name, String aValue) {
        addManualDialog.getjLabel2().setText(name);
        addManualDialog.getjTextField1().setText(aValue);
        confirm = false;
        addManualDialog.setVisible(true);
    }
    public String getValue() {
        return addManualDialog.getjTextField1().getText();
    }
    public boolean cheak() {
        if ("".equals(addManualDialog.getjTextField1().getText()))
{
            return false;
        }
        return true;
    }
    public boolean isConfirmed() {
        return confirm;
    }
    public void confirm() {
```

```

        confirm = true;
        addManualDialog.setVisible(false);
    }
    public void cancel() {
        confirm = false;
        addManualDialog.setVisible(false);
    }
}

```

Листинг 6 - файл <AddParametersLogic>

```

package guilogic;

import database.Executor;
import database.Queries;
import gui.AddParametersDialog;
import gui.MessageDialog;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JComboBox;

public class AddParametersLogic {

    private AddParametersDialog dialog;
    private Executor executor = new Executor();
    private DefaultComboBoxModel model;
    private ResultSet rs;
    private int idMobilephone, index;
    private String aValue;
    private boolean editing = false;

    public AddParametersLogic() {
        dialog = new AddParametersDialog(null, true, this);
    }
    public void initComboBox() {
        rs =
executor.getAllFromTable(Queries.SELECT_ALL_FROM_PARAMETERS_NAME);
        model = new DefaultComboBoxModel();
        try {
            while (rs.next()) {
                model.addElement(rs.getObject(1) + " " +
rs.getObject(2));
            }
        } catch (SQLException ex) {

```

```

        new MessageDialog(null,
true).showErrorMessage("Неудалось инициализировать комбобокс.");
    }
    dialog.getjComboBoxParametr().setEnabled(true);
    updateJComboBox(model);
}
public void initComboBox(int idParameter) {
    model = new DefaultComboBoxModel();
    model.addElement(idParameter + " " +
executor.getParamNameByID(idParameter));
    updateJComboBox(model);
    dialog.getjComboBoxParametr().setEnabled(false);
}
void updateJComboBox(DefaultComboBoxModel model) {
    dialog.getjComboBoxParametr().setModel(model);
}
private void addNewParameterName(String param) {
    index =
executor.getFreeId(Queries.SELECT_ALL_FROM_PARAMETERS_NAME);
    executor.insertParameter(index, param);
    model.addElement(index + " " + param);
    updateJComboBox(model);
}
public void confirm() {
    JComboBox jComboBox = dialog.getjComboBoxParametr();
    Object returned, selectedItem;
    returned = selectedItem = jComboBox.getSelectedItem();
    if (jComboBox.getSelectedIndex() == -1) {
        returned = cheack(selectedItem);
        if (Boolean.valueOf(returned.toString())) {
            addNewParameterName(selectedItem.toString());
            addParameter(index);
        }
    } else {
        if (!editing) {
addParameter(Integer.valueOf(returned.toString().split(" ")[0]));
        } else {
updateParameter(Integer.valueOf(returned.toString().split(" ")[0]));
        }
    }
    dialog.setVisible(false);
}
}

```

```

    public void cancel() {
        dialog.setVisible(false);
    }
    private void addParameter(int index) {
        if (!"".equals(dialog.getjTextFieldValue().getText())) {
            executor.insertMobilephoneParameter(idMobilephone,
index, dialog.getjTextFieldValue().getText());
        } else {
            new MessageDialog(null, true).showErrorMessage("Все
поля обязательны для заполнения.");
        }
    }

    private void updateParameter(int index) {
        if (!"".equals(dialog.getjTextFieldValue().getText())) {
            executor.updateMobilephoneParameter(idMobilephone,
index, dialog.getjTextFieldValue().getText());
        } else {
            new MessageDialog(null, true).showErrorMessage("Все
поля обязательны для заполнения.");
        }
    }
    public void showAddDialog(int idMobilephone) {
        this.idMobilephone = idMobilephone;
        editing = false;
        initComboBox();
        clearFields();
        dialog.setVisible(true);
    }
    public void showEditDialog(int idMobilephone, int idParameter,
String aValue) {
        this.idMobilephone = idMobilephone;
        this.aValue = aValue;
        this.editing = true;
        initComboBox(idParameter);
        initField();
        dialog.setVisible(true);
    }

    private Object cheack(Object selectedItem) {
        for (int i = 0; i < model.getSize() - 1; i++) {
            if
(name(model.getElementAt(i)).equalsIgnoreCase(selectedItem.toString())
) {

```

```

        return model.getElementAt(i);
    }
}
return "true";
}
private String name(Object selectedItem) {
    Scanner scan = new Scanner(selectedItem.toString());
    scan.next();
    return scan.nextLine().trim();
}
private void clearFields() {
    dialog.getjComboBoxParametr().setSelectedIndex(-1);
    dialog.getjTextFieldValue().setText("");
}
private void initField() {
    dialog.getjTextFieldValue().setText(aValue);
}
}

```

Листинг 7 - файл <MobilephonesDialogLogic>

```

package guilogic;

import database.Executor;
import gui.MessageDialog;
import gui.MobilephoneDialog;
import java.awt.Frame;
import java.sql.ResultSet;
import javax.swing.JTable;
import model.TableModel;

public class MobilephonesDialogLogic {

    private MobilephoneDialog mobilephonesDialog;
    private BuyPhoneLogic buyPhoneLogic = new BuyPhoneLogic();
    private FindPhoneLogic findPhoneLogic = new
FindPhoneLogic(this);
    private Executor executor = new Executor();
    ResultSet empty = executor.mobileParametersById(0);
    private boolean solded = true;

    public MobilephonesDialogLogic(Frame parent) {
        mobilephonesDialog = new MobilephoneDialog(parent, true,
this);
    }
}

```

```

    public boolean isSelled() {
        return sold;
    }
    public void findButton() {
        findPhoneLogic.show();
    }
    public void closeButton() {
        clearTableParameter();
        mobilephonesDialog.setVisible(false);
    }
    public void clearTableParameter() {
        mobilephonesDialog.getTable2().setModel(new
TableModel(empty));
    }
    public void buyButton() {
        JTable table = mobilephonesDialog.getTable1();
        int selectedRow = table.getSelectedRow();
        if (sold && selectedRow != -1) {
            int id = (int) table.getValueAt(selectedRow, 0);
            String model = (String) table.getValueAt(selectedRow,
1);
            String mark = (String) table.getValueAt(selectedRow,
2);
            int price = (int) table.getValueAt(selectedRow, 6);
            int count = (int) table.getValueAt(selectedRow, 7);
            buyPhoneLogic.showDialog(id, mark, model, price,
count);
            if (buyPhoneLogic.isSelled()) {
                showSelledPhones();
            }
        }
    }
    public void updateTable(ResultSet rs) {
        mobilephonesDialog.getTable1().setModel(new
TableModel(rs));
        mobilephonesDialog.setVisible(true);
    }
    public void selected() {
        int column;
        JTable table = mobilephonesDialog.getTable1();
        int selectedRow = table.getSelectedRow();
        if (selectedRow != -1) {
            column = sold ? 0 : 1;

```

```

        ResultSet resultSet =
executor.mobileParametersById((int) table.getValueAt(selectedRow,
column));
        if (resultSet == null) {
            new MessageDialog(null,
true).showErrorMessage("Неудалось считать данные из БД.");
        } else {
            mobilephonesDialog.getTable2().setModel(new
TableModel(resultSet));
        }
    }
    public void setVisible() {
        mobilephonesDialog.setVisible(true);
    }
    public void showPhones() {
        ResultSet resultSet = executor.findAllMobilephones(false);
        if (resultSet != null) {
            mobilephonesDialog.getTable1().setModel(new
TableModel(resultSet));
        }
        sold = true;
    }
    public void showSelledPhones() {
        ResultSet resultSet = executor.findAllSelledPhones();
        if (resultSet != null) {
            mobilephonesDialog.getTable1().setModel(new
TableModel(resultSet));
        }
        sold = false;
    }
}

```

```

package <settings>
Листинг 1 - файл <settings>
package settings;

```

```

import gui.MessageDialog;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Properties;
//Настройки подключения к БД.

```


//Класс синглтон, для того чтоб в программе не было нескольких объектов этого класса.

```
public class Settings {
    private static Settings instance;
    private Properties properties = new Properties();
    //Приватный конструктор
    private Settings() {
        restoreProperties();
    }
    //Получить объект класса, если не существует то создается.
    public static Settings getInstance() {
        if (instance == null) {
            instance = new Settings();
        }
        return instance;
    }
    //Стандартные настройки подключения.
    public void useDeffaultSetting() {
        properties.setProperty("ip", "localhost");
        properties.setProperty("port", "3306");
        properties.setProperty("dbname", "master");
        properties.setProperty("user", "root");
        properties.setProperty("password", "root");
    }
    //Восстановить настройки из файла.
    private void restoreProperties() {
        try {
            properties.load(new FileReader(new
File("connection.properties"))));
        } catch (IOException ex) {
            new MessageDialog(null,
true).showErrorMessage("Неудалось считать файл настроек.");
            useDeffaultSetting();
        }
    }
    //Сохранить настройки в файл.
    public void saveProperties() {
        try {
            properties.store(new FileWriter(new
File("connection.properties")), "Настройки подключения к БД.");
        } catch (IOException ex) {
            new MessageDialog(null,
true).showErrorMessage("Неудалось сохранить файл настроек.");
        }
    }
}
```

```
    }  
    public Properties getProperties() {  
        return properties;  
    }  
}
```