

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ

к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

на тему: "Системний аналіз процесу обслуговування клієнтів та розробка системи
інтернет-магазину товарів для велоспорту"

Виконав: студент групи

_____ *Грєшинов К. А.*
(підпис)

Керівник роботи:

К.Т.Н, доцент
(посада, науковий ступінь вчене звання)

_____ *Книрік Н.Р.*
(підпис)

м. Миколаїв – 2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра інформаційних управляючих систем та технологій

Спеціальність 124 "Системний аналіз"

Освітня програма "Системний аналіз"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

к.т.н, доц.НУК Приходько К.С.

" ____ " _____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття ступеня вищої освіти "бакалавр"

Студенту Грешнову Костянтину Андрійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи: "Системний аналіз процесу обслуговування клієнтів та розробка системи інтернет-магазину товарів для велоспорту"

Керівник роботи К.Т.Н, доцент Книрик Н. Р.

Затверджені наказом ректора № 243-уч від "18" березня 2024 року.

2. Термін подання роботи: 20 червня 2024 р.

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

4. Перелік питань, що належать до розробки (найменування розділів):

1. Аналіз предметної області і постановка задачі; 2. Розробка проекту автоматизованої системи; 3. Реалізація проекту автоматизованої системи; 4. Охорона праці.

5. Перелік презентаційних матеріалів:

1. Структура існуючої інформаційної системи; 2. Схема запропонованої інформаційної системи; 3. Діаграма варіантів; 4. Концептуальна модель даних інформаційної системи; 5. Логічна модель даних; 6. Діаграма класів системи; 7. Діаграма розгортання системи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1			
2			
3			
4			

7. Дата видачі завдання 05 лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області і постановка задачі	26.02.24	Виконано
2	Розробка проекту автоматизованої системи	18.03.24	Виконано
3	Реалізація проекту автоматизованої системи	10.05.24	Виконано
4	Охорона праці	26.05.24	Виконано
5	Оформлення ПЗ	17.06.24	Виконано
6			
7			
8			
9			
10			

Студент
(підпис)

_____ (ПІБ)

Грешнов К. А.

Керівник роботи
(підпис)

_____ (ПІБ)

Книрік Н.Р.

АНОТАЦІЯ

Дана дипломна робота присвячена системному аналізу процесу обслуговування клієнтів та розробці системи інтернет-магазину товарів для велоспорту. Метою розробки є підвищення ефективності роботи магазину і як наслідок зростання прибутку самого магазину.

У дипломній роботі формулюється загальна структура керування магазином, вимоги до системи, розробляються концепції системи і виконується вибір найкращої, розробляється технічне завдання. Далі проводиться розробка основних проектних рішень по технічним, інформаційним і програмному забезпеченню, формується робоча документація по програмному забезпеченню.

Дипломна робота викладена на 96 аркушах машинописного тексту, містить 9 рисунків, 4 додатка, список використаних джерел в кількості 10 найменувань

ABSTRACT

This thesis is devoted to a systematic analysis of the customer service process and the development of an online cycling store system. The aim of the development is to increase the efficiency of the store and, as a result, the profit of the store itself.

The thesis formulates the general structure of the store management, the requirements for the system, develops system concepts and selects the best one, and develops the terms of reference. Next, the main design solutions for hardware, information and software are developed, and working documentation for the software is created.

The thesis is set out on 96 pages of typewritten text, contains 9 figures, 4 appendices, a list of references in the amount of 10 titles

Ключові слова

Інтернет-магазин; Програмне забезпечення; Автоматизована система;
Системний аналіз; Підприємство; Веб-сайт; Магазин велотоварів;
Дослідження; Інформаційне забезпечення; Персональні дані.

Keywords

Online store; Software; Automated system; System analysis; Enterprise; Website; Bicycle shop; Research; Information support; Personal data.

ЗМІСТ

ВСТУП.....	9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ.....	10
1.1 Загальний опис проблеми.....	10
1.2 Аналіз підприємства для якого виконується автоматизація.....	15
1.3 Аналіз існуючих рішень.....	19
2 РОЗРОБКА ПРОЕКТУ АВТОМАТИЗОВАНОЇ СИСТЕМИ.....	24
2.1 Інформаційне забезпечення.....	24
2.2 Програмне забезпечення.....	28
3 РЕАЛІЗАЦІЯ ПРОЕКТУ АВТОМАТИЗОВАНОЇ СИСТЕМИ.....	34
4 ОХОРОНА ПРАЦІ.....	35
ВИСНОВКИ.....	44
ПЕРЕЛІК ПОСИЛАНЬ.....	45
ДОДАТКИ.....	46

Вступ

Дипломна робота присвячена системному аналізу процесу обслуговування клієнтів та розробці системи інтернет-магазину товарів для велоспорту, запропоновані різні методи для поліпшення ефективності роботи магазину, розроблена покращена система автоматизації робочого місця менеджера для більш стабільного і постійно зростаючого прибутку магазину.

Останні роки ознаменовані стрімким зростанням споживчого попиту. У зв'язку з цим з'явилася велика кількість точок, що здійснюють торгівлю велотоварами. 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 може послужити магазин товарів для велоспорту «ВелоБаза».

В даний час на фірмі продажів товару і пошуку клієнтів засновано в основному на паперовій підготовці пропозицій та їх розсилці, 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 комп'ютера;
- труднощі під час пошуку та ліквідації помилок, допущених на ранніх стадіях підготовки реклами.

У зв'язку з цим, застосування сучасних засобів проектування 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 проекту є розробка веб-сайту з функціями інтернет магазину для магазину товарів для велоспорту.

Дана розробка дозволить спростити та суттєво прискорити процес представлення фірми на ринку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Загальний опис проблеми

Відповідно до маркетингових досліджень, 85% рішень про покупку приймаються безпосередньо в магазині [5]. Покупець, перебуваючи в торговому залі, піддається впливу викладки, рекламних матеріалів, музики і здійснює незаплановані покупки, але в даний час набули широкого використання продажі через інтернет-магазини. Вони надають зручний спосіб вибору, замовлення і покупки товару не виходячи з дому [8]. У даному розділі ми опишемо і порівняємо ці два методи продажу товарів та ефективності двох абсолютно різних систем продажу.

У першому випадку коли продаж здійснюється безпосередньо в самому магазині і покупець, знаходиться в торговому залі, сучасні ритейли застосовують різні інструменти для збільшення часу перебування споживача в магазині і стимулювання його купівельної активності.

У першому випадку для продажу даного товару необхідно виконати дуже великий перелік заходів для того щоб залучити покупця і на підсвідомому рівні змусити його купити товар [7]:

1. Забезпечення чистоти.

Перше, на що звертає увагу споживач – це чистота підлоги в торговому залі, відсутність пилу на полицях і товарах, охайність продавців і консультантів. Тому перш ніж здійснювати розстановку торгового устаткування і товарів, треба забезпечити чистоту. Уявіть себе на місці споживача: ви заходите в магазин, вибираєте товар, а на ньому шар пилу. Напевно, у вас сформується негативне ставлення до магазину і з'явиться відчуття, що споживача тут не люблять.

2. Забезпечення огляду всього товару для споживача. Споживач, заходячи в торговий зал, повинен відразу зрозуміти, куди йому треба йти, щоб

купити все необхідне. Для цього в торговому залі розміщуються покажчики товарних секцій і забезпечується максимальна візуалізація товарів.

3. Доступність товару для споживача. Товар повинен бути максимально доступний покупцеві. Під доступністю розуміється, перш за все, можливість взяти товар, не докладаючи для цього якихось серйозних зусиль. Необхідно також враховувати споживче призначення товару. Якщо це товар для дітей, наприклад, іграшка, значить, його треба розмістити на рівні зростання дитини.

4. Комфортність присутності в магазині. Поняття комфорту дуже широке, і його сутність багато в чому залежить від індивідуальних особливостей людини, його психографічних характеристик.

Проте можна визначити ряд стандартних вимог до забезпечення комфорту для основної сукупності споживачів. До них можна віднести наступне:

1. Зручність переміщення в торговому залі. Людина повинна вільно пересуватися по товарних секціях, не натикаючись на інших людей і не зачіпаючи торговельне обладнання.

2. Музика в торговельному залі. Музичний супровід процесу покупок – ефективний засіб для стимулювання продажів. Згідно з результатами маркетингових досліджень, динамічна музика обумовлює активне пересування по торговому залу і прискорення процесу прийняття рішення про покупку, збільшення імпульсних покупок. Повільна (найчастіше класична) музика розслабляє покупця, забезпечує більш тривале його присутність в торговому залі. Її використовують у торговельних залах, де представлені товари ретельного вибору, дорогі товари. Сучасні ритейли здійснюють підбір музичних творів, найбільш придатних позиціонуванню магазину, а також поведінкових особливостей покупця.

3. Якісне обслуговування. Персонал магазину повинен бути готовий надати допомогу споживачеві в будь-який момент. Ввічливість продавців і консультантів, здатність надати всю необхідну інформацію високо оцінюється покупцями і формує лояльність до магазину. Якісне обслуговування

передбачає також уважне ставлення до потреб покупця. Наприклад, якщо в магазин зайшла жінка з маленькою дитиною, то хороший консультант напевно допоможе їй розмістити дитячу коляску і максимально швидко купити все, що їй необхідно.

4. Надання додаткових послуг (розважальних, інформаційних та інших). Найчастіше покупець приходить у магазин не тільки для того, щоб зробити покупки, але і для отримання консультації про продукт або способах його використання або, наприклад, для того, щоб оплатити через автомат послуги стільникового зв'язку. Саме тому в магазині повинні бути необхідні атрибути, що дозволяють задовольнити відповідні потреби.

5. Наявність допоміжного обладнання, що полегшує процес покупки - купівельних корзин, каталог, стільців для відпочинку, підставок для сумок, спеціальних місць для розваги і відволікання малюків, які супроводжують своїх матерів.

6. Забезпечення заповнення полиць. Важливо сформувати у споживача враження постійної наявності всього необхідного йому асортименту в магазині. Тому полки завжди повинні бути заповнені товарами, що задовольняють потребу споживачів.

7. Забезпечення інформативності товарів. Упаковка товару повинна містити всю необхідну споживачеві інформацію.

8. Ціна повинна бути зрозуміла споживачеві.

Весь цей величезний перелік заходів призводить до збільшення часу передпродажної підготовки і до подорожчання товару, тому розглянемо другий спосіб продажу товарів нашої галузі, продаж велотоварів через інтернет-магазини.

Робота менеджерів магазинів ускладнена великою кількістю товарів, замовлень. Потрібно швидко опрацювати замовлення, перевірити наявність товару. Необхідно постійно оновлювати товари, видаляти не потрібні, редагувати описи і т.д. — це займає велику кількість часу і потребує великих

зусиль [10]. Тому інформаційна система, що розробляється призначена спростити роботу магазину для продажу товарів для велоспорту.

У завдання адміністратора магазину входить надання клієнтам магазину інформації про пропоновані товари, прийом замовлень клієнтів, розрахунки з постачальниками товарів [3, 9].

Обґрунтування актуальності задачі що розв'язується: як вже зазначалося у вступі, сьогодні в торговельній діяльності широко застосовуються різні засоби автоматизації – від звичайних баз даних до радіо-ідентифікаційних міток. Хоча в нашій країні подібна практика поки що не дуже поширена, але певний прогрес все ж помітний – деякі фірми займаються розробкою програмного забезпечення створюють і продукти для автоматизації процесів у сфері торгівлі, які знаходять визнання не тільки в нашій країні, але і за її межами.

Використання Інтернету в торговельній діяльності також досить перспективний напрямок, тому що дозволяє за рахунок економії коштів істотно підвищити рівень доходів [8]. До того ж подібний вид торгівлі дуже зручний і для покупців, хоча реалізація окремих товарів затруднена. Уся необхідна до роботи інформація, а точніше всі дані будуть утримуватися у спеціальних таблицях. Усі таблиці є пов'язаними між собою, в них виконуються всі вимоги, запропоновані до реляційних баз даних [4, 6]. Клієнт при замовленні товару може вибирати між різними службами доставки (з різними термінами доставки і платою), і навіть між різними формами оплати. У СУБД ведеться облік всіх постачальників, тобто обсягів поставлених ними товарів у грошовому і кількісному вираженні. Також підраховуються обсяги виконаних доставок різними службами. Для кожного клієнта розраховується знижка в залежності від того, на яку суму їм було зроблено покупку. Також кожному з клієнтів відкривається персональний грошовий рахунок, який можна поповнювати та з якого можна оплачувати покупки.

Цей проект спростить внесення й зміну всіх персональних даних. Він зробить процес покупок клієнтів максимально зручним і зрозумілим, а

обслуговування для адміністраторів простим й єдино правильним. Згідно з інформаційним пошуком виявилось, що динамічне, конкурентне бізнес-середовище, що постійно змінюється, вимагає швидкого засвоєння можливостей, які надаються менеджеру з персоналу новими інформаційними технологіями [3, 9]. Менеджери з персоналу виконують таку роботу й утворюють такий продукт, що неможливо скопіювати, запозичити або купити як будь-яку іншу технологію. Вони створюють корпоративний дух, творче середовище, ефективну структуру організації – саме ці суттєвості і є основною цінністю компанії-підприємства. До найбільш спеціалізованих функцій відділу кадрів, незалежно від його статусу, узагальнено можна віднести динамічне управління якісною і кількісною мірою персоналу підприємства [9]. Із цим узагальненим формулюванням пов'язана структурована сукупність комплексів завдань, частина з яких може ефективно підтримуватися новими інформаційними технологіями. Ефективність функціонування системи управління персоналом на підприємстві в значній мірі залежить від якості та кількості функціонуючої інформації, інформаційних баз підприємства.

Інформаційні бази підприємства є сукупністю даних про стан керованої системи і тенденції її розвитку за допомогою кількісних характеристик процесів, що відбуваються у складі працівників та в їх виробничому і соціальному середовищі. Власне, будь-яке управлінське рішення повинне ґрунтуватися на знанні ситуації, впливаючих чинників, їх взаємозв'язків. Інформація сприяє реалізації всіх функцій управління: планової, організаційної, контрольної, регулюючої, розподільної.

Багато процесів, що визначають кадрову ситуацію, вельми динамічні: змінюються чисельний склад персоналу у зв'язку з текучістю, його структура по всій сукупності ознак: віку, стажу роботи і т.д., постійно йде процес внутрішнього руху кадрів між структурними підрозділами, міняються професії, рівень кваліфікації. Інформація в даному випадку виступає як засіб контролю за відповідністю фактичної структури робочої сили тій, що вимагається, за своєчасністю укомплектування вакантних робочих місць.

Тому основними вимогами, що пред'являються до кадрової інформації, є оперативність і надійність [9].

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 [9]. 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 такими 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 веломагазині задіяні наступні працівники: відповідальний за магазин - директор, у нього в підпорядкуванні відділ закупівель, відділ продажу, адміністратор сайту 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 сайту займається рекламою в мережі Інтернет, бухгалтер веде бухгалтерський облік магазину, Директор слідкує за тим щоб діяльність фірми була налагоджена найкращим чином [7, 9].

Організаційна мережа магазину товарів для велоспорту представлена нижче.

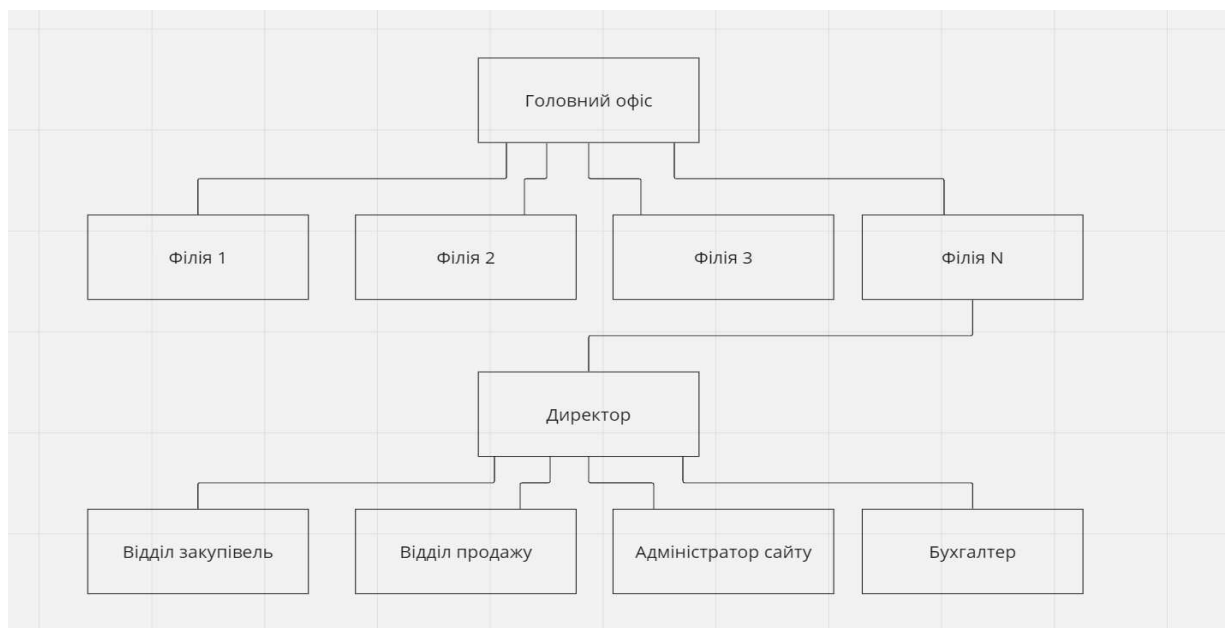


Рисунок 1.1 – Організаційна мережа магазину товарів для велоспорту

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;
4. Виведення ціни товару;
5. Замовлення вибраного товару;
6. Оформлення замовлення.

1.2.1 Структура існуючої інформаційної системи

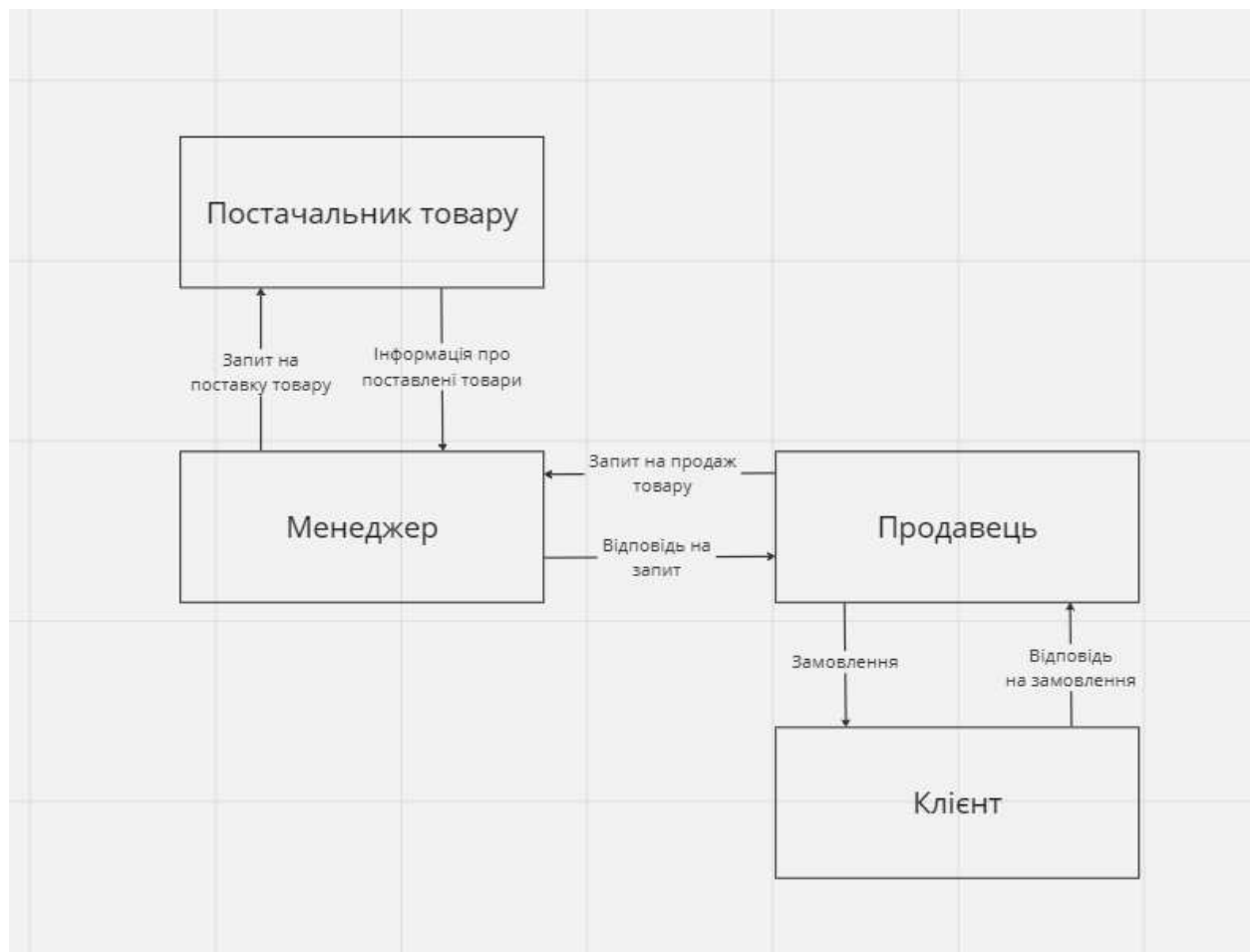


Рисунок 1.2 - Структура існуючої інформаційної системи

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 пропонуємої інформаційно-управляючої системи представлена на Рисунку 1.3

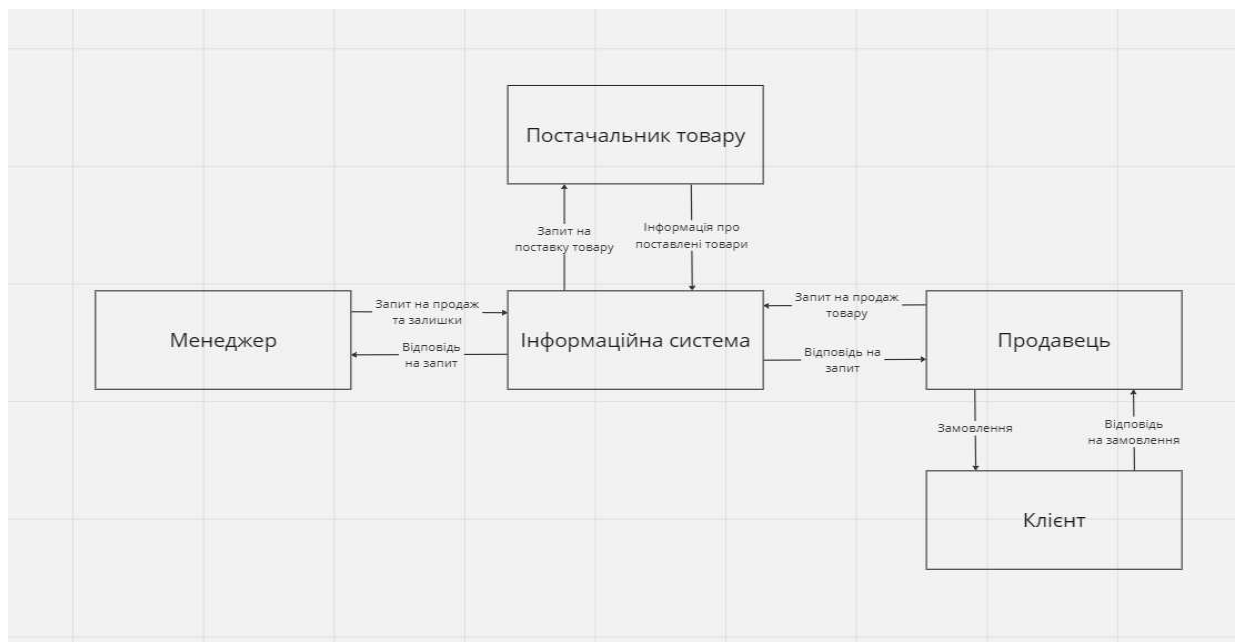


Рисунок 1.3 – Схема запропонованої системи

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 проведених досліджень об'єкта автоматизації були з'ясовані основні етапи, які необхідно реалізувати, це подання підприємства в мережі Інтернет (інформація, послуги, вартість робіт, виконані роботи і т.д.), формування замовлень за допомогою спеціальної форми, на підставі замовлень підготовка відомостей покупних матеріалів.

Розглянемо різні концепції вирішення поставленого завдання. На даний момент існує три реальні шляхи:

- купити готовий комплекс управління сайтом і впровадити його на підприємстві
- залучити сторонніх фахівців в області розробки Web-сайтів
- спроектувати веб-сайт, що відповідає поставленим вимогам за допомогою штатних фахівців компанії. Як правило даним завданням займається відділ АСУП.

Розглянемо кожен з цих концепцій для вибору найбільш оптимальної. На сьогоднішній день існує досить багато систем управління сайтами. Дані системи представлені як зарубіжними, так і вітчизняними виробниками.

1.3.1 Система управління сайтом (контентом)

Управління змістом сайту надає адміністратору (власнику) веб-сайту всебічні можливості по формуванню його структури. Можливо самостійно змінювати і додавати інформацію на сайт - нові статті, тексти, малюнки, створювати нові розділи і змінювати вже існуючі - і все це доступно і легко. Таким чином система управління сайтом CMS ідеально підходить для власників веб-ресурсів, які бажають ефективно, зручно і швидко управляти змістом сайту і його структурою. Власник сайту зможе самостійно додавати, редагувати, видаляти сторінки і розділи на сайті, без знання спеціальних технологій. Всі ці дії здійснюються в захищеній адміністративній частині сайту, у вікні звичайного браузера.

Адміністративна панель CMS системи управління веб-сайтом дозволяє міняти або додавати нову інформацію для різних мовних версій веб-сайту. Зміни на сайті відображаються відразу ж після їх внесення і збереження.

Системи управління веб-сайтом представляють собою інструменти, які дозволяють моделювати розгалужені структури сайтів і керувати їхнім інформаційним наповненням. Такі системи управління контентом сайту не потребуватиме спеціальних технічних навичок, таких як програмування або HTML-верстка. Робота системи нагадує роботу текстового редактора Windows або інших редакторів веб-сторінок. Досить виділити текст, скопіювати і вставити. Впровадження системи управління веб-сайтом дозволяє забезпечити контроль над доступом до сайту і внесенням змін.

Базова система управління сайтом включає такі можливості:

- Пошук інформації на сайті
- Збір даних про клієнтів та потенційних клієнтів

- Формування та редагування опитувань
- Аналіз відвідування веб-сайту

Основні модулі системи управління вмістом сайту:

- Статті - додавання, редагування, планування і опублікування "статей" (сторінок сайту) в WORD-подібному інтерфейсі.
- Пункти меню - додавання, редагування, управління пунктами меню сайту будь-якого рівня.
- Новини - додавання, редагування, і опублікування новин.
- Фотогалерея - можливість робити фотогалереї з подгалереями, автоматичне масштабування фото.
- Налаштування - тут зберігаються всі налаштування сайту та системи управління сайтом.
- Користувачі - управління правами користувачів, зареєстрованих на сайті.
- Каталог фірм - додавання, редагування, опублікування фірм в підгрупах будь-якої складності.
- Опитування - додавання, редагування опитувань на сайті. Результати представляється у вигляді графіків. Кількість питань: від двох до десяти. Використання системи керування вмістом сайту не вимагає встановлення спеціального програмного забезпечення на робочому місці. Для редагування й адміністрування використовується звичайний браузер. Інтуїтивний інтерфейс і простота роботи із системою дозволяє полегшити управління сайтом і знизити подальші витрати на підтримку веб-ресурсу. Для роботи з системою управління необхідно мати лише невеликі навички роботи в Інтернеті.

Адміністративна панель CMS системи управління веб-сайтом дозволяє міняти або додавати нову інформацію для різних мовних версій веб-сайту. Зміни на сайті відображаються відразу ж після їх внесення і збереження.

Можливості адміністрування CMS:

- Можливість створювати необмежену кількість сторінок
- Чітка організація структури сайту

- Для кожної динамічної сторінки можна створити свій опис і ключові слова з метою підвищення рейтингу в пошукових системах
- Початок і закінчення публікації будь-яких матеріалів можна запрограмувати за календарем;
- Можливість обмежити доступ до певних розділів сайту тільки для зареєстрованих користувачів;
- Завантаження зображень за допомогою браузера в вашу власну бібліотеку;
- Для подальшого використання з будь-якого місця сайту;
- Опитування та голосування для ефективного зворотного зв'язку;
- Різні модулі - такі як останні новини, лічильник відвідувань, докладна статистика відвідувань, гостьова книга, форум і т.д.;
- Можливість створення не однієї, а декількох форм зворотного зв'язку для кожного контакту;
- Зміна порядку об'єктів, включаючи новини, питання, статті;
- Генератор показу випадкової новини;
- Модуль прийому від авторів новин, статей і посилань;
- Ієрархія об'єктів - кількість секцій, розділів, підрозділів і сторінок, залежить від вашого бажання;
- Бібліотека зображень дозволить зберігати всі файли під рукою для легкого доступу;
- Менеджер розсилки новин;
- Менеджер архіву;
- Можливість роздрукувати або відправити на e-mail будь-яку статтю з сайту;
- Вибір з 3 візуальних редакторів, що спрощує редагування матеріалів до рівня редагування тексту в програмі Word;
- Можливість легкої зміни дизайну;
- Економне використання місця на сервері за рахунок використання бази даних;
- Можливість використання адрес сторінок адаптованих для кращої індексації всіма пошуковими системами.

За допомогою системи управління веб-сайтом можна створювати:

- Корпоративні веб-сайти та портали;
- Онлайн-магазини;
- Невеликі сайти;
- Сайти для некомерційних організацій, шкіл і т.д.;
- Персональні сторінки;
- Сайти спільнот;
- Онлайн-журнали, газети та ін.

2 РОЗРОБКА ПРОЕКТНИХ РІШЕНЬ

2.1 Інформаційне забезпечення

2.1.1 Вибір засобів моделювання

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

Головна особливість індустрії сучасних корпоративних програмних комплексів складається в концентрації складності на початкових етапах аналізу вимог і проектування 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 проектування, 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

(декларуючи первинність функціонального або інформаційного моделювання), по типу 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 товарів для велоспорту

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

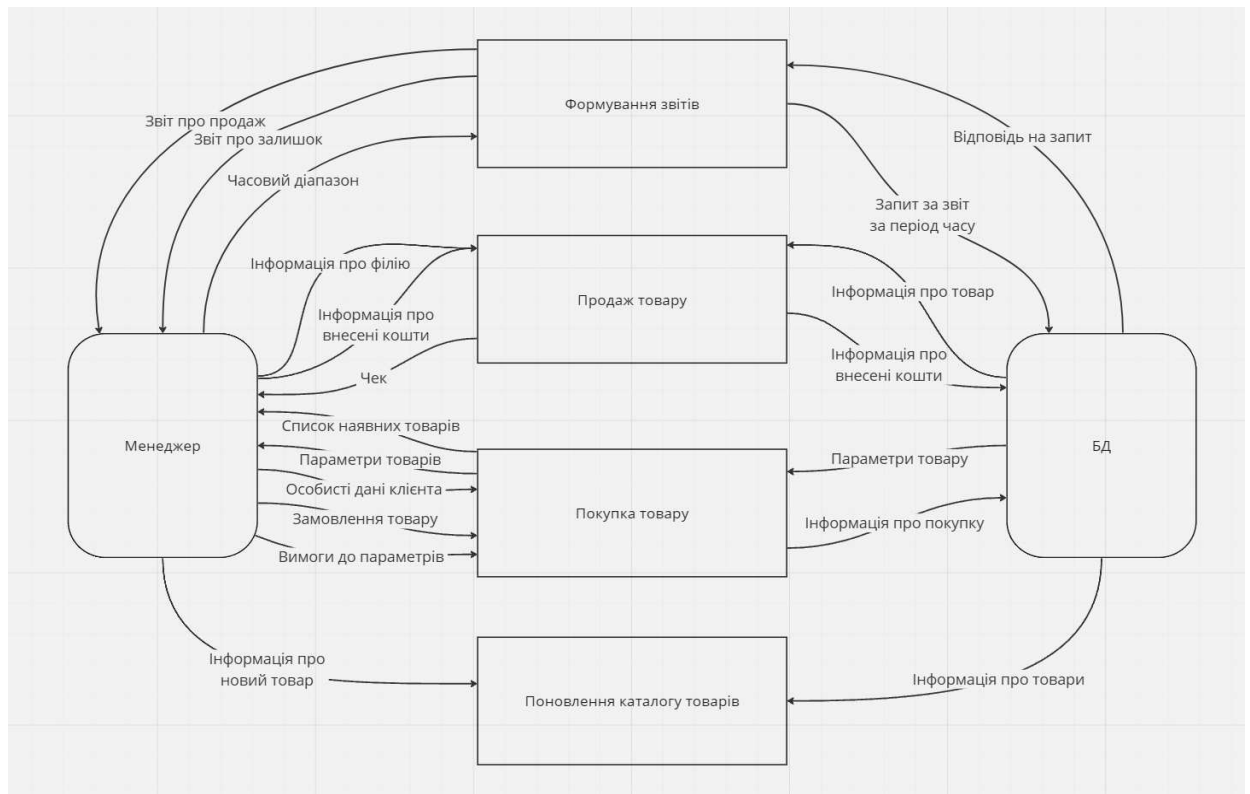


Рисунок 2.1 – Діаграма варіантів використання інформаційної систему магазину спортивних товарів.

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 обраної 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 магазину товарів для велоспорту.

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.



Рисунок 2.3 – Концептуальна модель даних

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 товарів для велоспорту з деталізацією атрибутів показана на рисунку 2.4.

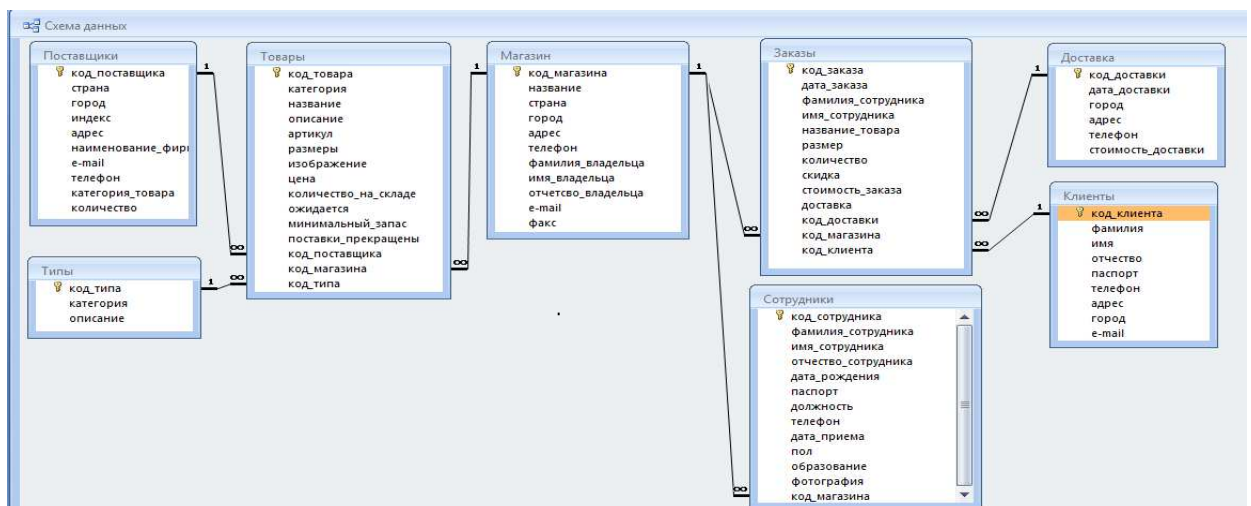


Рисунок 2.4 – Логічна модель даних

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.

2.2 Програмне забезпечення

2.2.1 Вибір мови програмування

Виходячи з результатів дослідження середовища виконання програмного забезпечення розробки АРМ менеджера інформаційної системи магазину мобільного зв'язку, можна сформулювати наступні вимоги до мови програмування:

- незалежність від архітектури;
- робота в мережевому середовищу;
- робота з СУБД;
- гнучкість щодо внесення майбутніх змін.

Приймаючи до уваги наведені вимоги, для вирішення поставленого завдання була обрана мова, що була розроблена програмістами Sun Microsystems, - Java.

Фірма Sun Microsystems визначає мову програмування Java наступним образом:

- об'єктно-орієнтований;
- простий;
- розподілений;
- стійкий;
- надійний;
- архітектурно-нейтральний;
- переносимий;
- інтерпретуємий;
- багатопоточний;
- динамічний.

Java орієнтований для застосування у мережі Internet/intranet має бібліотеку для роботи з протоколами TCP/IP, FTP и HTTP. Програми, які створені на мові Java, спілкуються з мережевими URL об'єктами так же просто, як з об'єктами локальної файлової системи. Сучасні версії мови Java пристосовані для створення серверних WWW програм в корпоративних системах.

Компілятор Java генерує об'єктний байт-код, якій не залежить від операційної платформи середовища виконання програми. Один й той самий байт-код може бути виконаний в різних операційних системах на різних апаратних засобах. Байт-код інтерпретується в машинний код даної платформи одразу за допомогою віртуальної Java машини (JVM). Слід зазначити той факт, що мові Java вже більше 25 років і у своєму розвитку вона перетерпіла певні зміни, що відбилися на її синтаксисі та бібліотеках. Тому необхідно вказати яка версія мови була використана при розробки програмного забезпечення інформаційної системи управлінського обліку. При розробці програми була обрана версія - Java 17.

2.2.2 Методи та засоби розробки

Для розробки діаграм була використана наступна програма ErWin 2020 R2.

CA ERwin Data Modeler - CASE-засіб для моделювання реляційних баз даних.

ERwin - засіб розробки структури бази даних (БД). ERwin поєднує графічний інтерфейс Windows, інструменти для побудови ER-діаграм, редактори для створення логічного та фізичного опису моделі даних і прозору підтримку провідних реляційних СУБД і настільних баз даних. За допомогою ERwin можна створювати або проводити зворотну розробку баз даних. Процес побудови інформаційної моделі в ERwin складається з наступних кроків:

- визначення сутностей;
- визначення залежностей між сутностями;
- завдання первинних та альтернативних ключів;
- визначення атрибутів сутностей;
- приведення моделі до необхідному рівню нормальної форми;

- перехід до фізичного опису моделі: призначення відповідностей «ім'я сутності - ім'я таблиці», «атрибут сутності - атрибут таблиці»;
- завдання тригерів, процедур і обмежень;
- генерація бази даних.

ERwin створює візуальне представлення (модель даних) для розв'язуваної задачі. Це подання може використовуватися для детального аналізу, уточнення і поширення частини документації, необхідної в циклі розробки. Також ERwin автоматично створює базу даних (таблиці, індекси, збережені процедури, тригери для забезпечення посилальної цілісності та інші об'єкти, необхідні для управління даними).

Рекомендується активно використовувати можливість спадкування класів, щоб спростити розмір кінцевого коду і тим самим знизити кількість ймовірних помилок.

2.2.3 Структура програмного забезпечення.

Вхідними даними для даного етапу проектування будуть моделі сутностей, а результатом стане модель специфікації проектних конструкцій розроблювального програмного забезпечення. У відповідності з прийнятим у для даної роботи підходом об'єктно-орієнтованого аналізу і проектування виконаємо подальшу специфікацію проектних рішень у виді діаграм класів. Класи визначають стан пропонованої програмної системи (через значення своїх атрибутів) і, значною мірою, її поведінку (через операції чи методи класів). Діаграма бізнесів-сутностей приведена на рисунку 2.6.

Сформульовані користувацькі вимоги тепер приймаємо за обмеження.

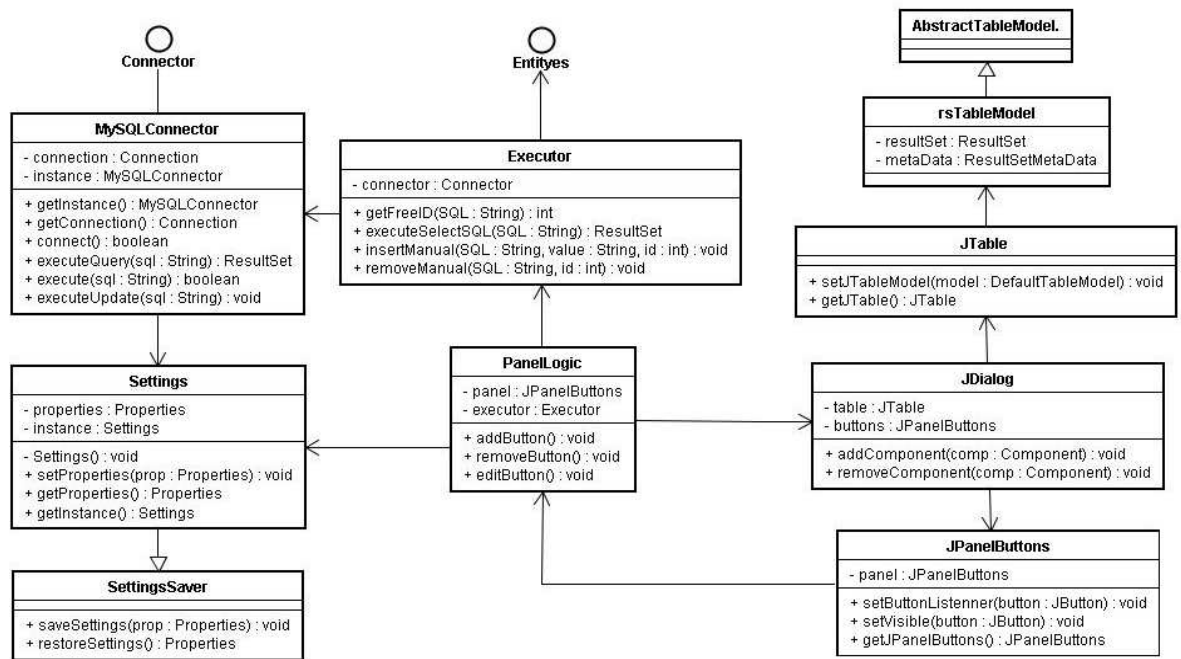


Рисунок 2.6 – Діаграма класів системи

Вкажемо 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 2.7

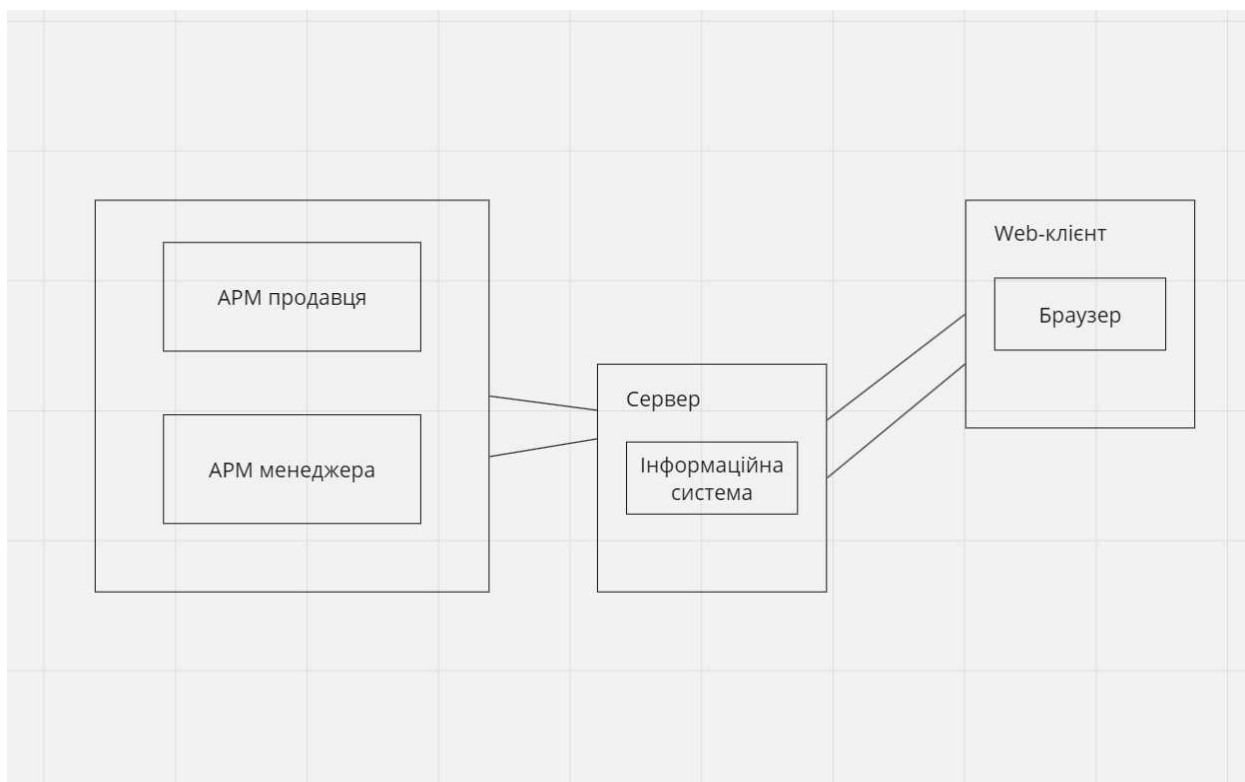


Рисунок 2.7 – Діаграма розгортання системи

3 РЕАЛІЗАЦІЯ ПРОЕКТУ АВТОМАТИЗОВАНОЇ СИСТЕМИ

У даній дипломній роботі була розроблена інформаційна система магазину товарів для велоспорту, яка призначена для автоматизації процесу продажу товарів, формування звітів про реалізований та замовлений товар, редагування даних і своєчасний перегляд нових каталогів, та ведення обліку.

Відповідно вимогам сформульованим у технічному завданні, яке приведено в додатку А, дана система виконує функції:

1. Додавання товарної позиції.
2. Редагування товарної позиції.
3. Внесення додаткової інформації про товар.
4. Внесення інформації про продаж товару.
5. Пошук по мережі магазинів до загальної бази товарів.
6. Формування звіту про продані товари.

Всі функції були реалізовані в відповідних модулях об'єктів системи. В якості результатів роботи даної підсистеми, користувач отримує необхідну йому інформацію.

При роботі з програмним забезпеченням системи варто керуватися документами «Керівництво користувача» (Додаток Б), «Технічний опис» (Додаток В).

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 додержуватися 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) встановлення 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 пов'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 працівників призводить 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, впроваджувати нові ефективні засоби забезпечення безпеки праці.

4.1 Аналіз небезпечних і шкідливих факторів в офісному приміщенні

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 комп'ютера супроводжується виникненням ряду шкідливих і небезпечних виробничих факторів фізичного і психофізіологічного характеру. Працівники, що обслуговують комп'ютер, 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 перевищують гігієнічні нормативи призводить до функціональних змін з боку нервової, серцево-судинної, нейрогуморальної та інших систем організму.

Фізичні перевантаження найчастіше торкаються зорової системи або кістково-м'язову систему (плечовий суглоб, шия, верхня частина спини). У цілому користувачі комп'ютера не зіштовхуються з якимось особливим психічним стресом і емоційними проблемами, дослідження показують одиничні випадки дуже напружений, зухвалий стрес, роботи на комп'ютері. Проте, рекомендується вживати заходів до запобігання надмірного дискомфорту і стомлення працюючих на комп'ютері. У більшості користувачів комп'ютерів виникає кістково-м'язовий дискомфорт через статичну напругу м'язів, незручної пози або часто повторюваних одноманітних рухів, ймовірність виникнення стійких порушень дуже мала, однак монотонні рухи при роботі на комп'ютері можуть викликати хронічні захворювання.

Важливим фактором надійної роботи засобів ВТ і високої працездатності обслуговуючого персоналу є кліматичні умови. Під кліматичними умовами виробничого середовища 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 комп'ютери 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 огороджують).

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 комп'ютера, 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 комп'ютера, знижуються ізоляційні властивості повітря. Перераховане вище обумовлює підвищені вимоги до кліматичних умов середовища в приміщеннях з комп'ютерами.

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 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 електромагнітного випромінювання

Для зниження впливу електромагнітного випромінювання треба застосовувати сертифіковані монітори. Монітори, які відповідають останнім стандартам TCO Certified та FCC є найбільш безпечними для здоров'я людини. Вони визначають максимально припустимі величини випромінювання магнітного й електричного полів, а також методи їхнього виміру. Помітимо, що стандарти FCC 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.

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 весь час були сфокусовані на одній відстані, працюючи з текстом, рекомендується встановлювати великий шрифт. Дуже важливо, щоб екран монітора не мерехтів. Частота оновлення зображення повинна бути не менше 75 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 в бік вікна, оскільки інтенсивна освітленість області зору може затопити потоками світла очі і розмити зображення на сітківці. Якщо приходить сидіти поруч з вікном, то треба розташовуватися під прямим кутом до нього, причому екран дисплея повинний бути перпендикулярний шибці - цим виключаються відблиски на екрані.

Форма спинки крісла повинна повторювати форму спини працюючого. Крісло необхідно встановити 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

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

Клавіатуру необхідно встановити так, щоб не треба було до неї тягнутися і нахилитися вперед. При зміні положення тіла (наприклад, з вертикального на похиле) обов'язково треба перемінити положення клавіатури і дисплея. Може виявитись корисною регульована підставка клавіатури, завдяки якій можна без усякої напруги працювати з комп'ютерною мишею. 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 встановити 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.

ВИСНОВКИ

В результаті виконання роботи була розроблена система обслуговування клієнтів та розробка системи інтернет-магазину товарів для велоспорту, який дозволяє спростити роботу з документообігом. Для досягнення поставленої мети були вирішені наступні задачі:

- досліджено предметну галузь програмного забезпечення;
- сформовано вимоги до системи обслуговування клієнтів Internet-магазину спортивних товарів;
- розроблено проект системи обслуговування клієнтів інтернет магазину товарів для велоспорту, який включає в себе три рівня представлення: ескізний проект; технічний проект та робочий, застосовуючи при цьому всі існуючі аспекти для розуміння проекту, а саме структуру програми; структури даних; потоки даних; керування; стани програми при виконанні; потоки та інтерфейси їх вводу та виводу.
- спроектовано яким чином користувач буде взаємодіяти з системою, яким повинен бути інтерфейс та його застосування в різних випадках використання.

ПЕРЕЛІК ПОСИЛАНЬ

1. Шілдт Г. Java. Повне керівництво 1-ий том: Видавничий дім «Діалектика», 2020. - 730 с.
2. Шілдт Г. Java. Повне керівництво 2-ий том: Видавничий дім «Діалектика», 2020. - 780 с.
3. Друкер П. Ефективне управління підприємством: «Вільямс», 2018. – 224с.
4. Новіков Б. А. Горшкова В. А. Графеева Н. Г. Основи технологій баз даних. Друге видання: Print2print, 2020, - 582с.
5. Полторак В., Тараненко І., Красовська О. Маркетингові дослідження: Центр навчальної літератури, 2017. – 342с.
6. Дейт К. Дж. SQL і реляційна теорія. Як грамотно писати код на SQL: Символ, 2010. – 480 с.
7. Гоффман Р., Коен Д., Тріфф Д. Майстри масштабування. Неочевидні істини для найуспішніших підприємців світу: Лабораторія, 2023. – 320с.
8. Гупта С. Цифрова стратегія. Посібник із переосмислення бізнесу: КМ-Букс, 2020. – 320с.
9. Горовіц Б. Ви те, що ви робите. Як створити корпоративну культуру: Наш Формат, 2020. – 224с.
10. Кучеренко Д. Менеджмент. Курс лекцій. 2-ге видання: Центр навчальної літератури (ЦНЛ), 2020. – 184с.

ДОДАТОК А – Технічне завдання

Призначення розробки системи

Назва програмного продукту: “Системний аналіз процесу обслуговування клієнтів та розробка системи інтернет-магазину товарів для велоспорту”. Програмний продукт призначений для автоматизації товаро–фінансових робіт підприємства.

Експлуатаційне призначення: при покупці товару, покупець повинен зареєструватися. Реєстрація проходить шляхом внесення особистих даних (ПІБ, телефон, адреса) в анкету з метою встановлення гарантійного строку для товару, 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 обіг 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 найму, 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 комп'ютером. До роботи із системою допускаються тільки користувачі, що ознайомлені з посібником користувача.

Вимоги до експлуатації, технічному обслуговуванню і збереженню

До роботи допускаються тільки підготовлені користувачі, що пройшли інструктаж з техніки безпеки роботи з комп'ютером і ознайомлені з посібником користувача. При виникненні збоїв у роботі, користувач повинен сповістити про це адміністраторів 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 комп'ютером і пожежної безпеки в робочому приміщенні.

Порядок контролю і приймання системи

Для контролю й прийому повинно бути надано керівництво оператора 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 з 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 упродовж 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 «Сохранить» і попадаємо в головне вікно програми. Навігаційні кнопки та панелі достатньо оснащені коментарями для розуміння та комфортної роботи з програмою.

На головному вікні зображені чотири кнопки: при натисканні на кнопку «Продажа товаров» відкривається вікно і ми можемо прослідкувати як продаються товари, за допомогою кнопки «Редактирование товаров» при натисканні відкривається вікно редагування та додавання товарів і наступна кнопка «Настройки» при натисканні на цю кнопку відкриваються налаштування БД.

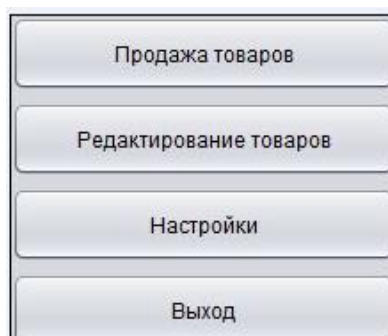


Рисунок Б1 – Головне вікно програми

В цьому вікні ми бачимо головне вікно продажу товарів.



це вікно для пошуку товару по певним параметрам;



це таблиця допустимих до продажу товару;



це таблиця проданих товарів;



оформити продаж товару;



Вихід.

ДОДАТОК В – Технічний опис

1 Призначення системи

Розроблена інформаційна система призначена адміністратора online-магазину товарів для велоспорту 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). Для роботи в діалоговому режимі використовується екран дисплея, клавіатура та комп'ютерна миша. Вхідні дані зберігаються на твердому диску комп'ютера. Програма працює під управлінням операційної системи Windows 7/8/10.

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.

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 “Налаштування підключення до БД”

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

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

Модуль “Продаж 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 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 >
```

Листинг 1 – файл <Connector>

```
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
        ProgressDialog(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(Queryes.SELECT_ALL_FROM_MOBILEPHONES + "
ORDER BY `idMobilephone`");
            } else {
                resultSet
connector.executeQuery(Queryes.SELECT_ALL_FROM_MOBILEPHONES + "
AND m.`count` > 0 ORDER BY `idMobilephone`");
            }
        } catch (SQLException ex) {
            new
                ProgressDialog(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(Queryes.SELECT_MOBILEPHONE_BY_ID);
            statement.setInt(1, idMobilephone);
            resultSet = statement.executeQuery();
        } catch (SQLException ex) {
            new
                ProgressDialog(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
                ProgressDialog(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
                ProgressDialog(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
        ProgressDialog(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
        ProgressDialog(null,
true).showErrorMessage("Неудалось изменить таблицу телефонов.");
    }
}

//Получить в свободный 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
        ProgressDialog(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
        ProgressDialog(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
        ProgressDialog(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(Queries.INSERT_INTO_PRODUCT);
        statement.setString(1, mark);
        id = getFreeId(Queries.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
        ProgressDialog(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(Queryes.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
                ProgressDialog(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(Queryes.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
                ProgressDialog(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
                ProgressDialog(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
                ProgressDialog(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
                ProgressDialog(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(Queries.INSERT_INTO_MOBILEPHONE_PARAMETERS);

        statement.setString(1, text);
        statement.setInt(2, idMobilephone);
        statement.setInt(3, index);
        statement.execute();
    } catch (SQLException ex) {
        new
        ProgressDialog(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(Queries.UPDATE_MOBILEPHONE_PARAMETER
);

        statement.setString(1, text);
        statement.setInt(2, idMobilephone);
        statement.setInt(3, index);
        statement.executeUpdate();
    } catch (SQLException ex) {
        new
        ProgressDialog(null,
true).showErrorMessage("Неудалось добавить параметр телефона.");
    }
}

//Удалить продажу.
public void removeOrder(int idOrder) {
    Connection connection = connector.getConnection();
    try {
        PreparedStatement statement =
connection.prepareStatement(Queries.DELETE_ORDER_BY_ID);
        statement.setInt(1, idOrder);
    }
}

```

```

        statement.execute();
        try {
            removeProductByIdMobilephone(idOrder,
connection);
        } catch (SQLException ex) {
        }
    } catch (SQLException ex) {
        new
            ProgressDialog(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
            ProgressDialog(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
        ProgressDialog(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.executeUpdate();
        } catch (SQLException ex) {
            new
            ProgressDialog(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 removeFilllial(int index) {
    try {
        Connection connection = connector.getConnection();
        PreparedStatement statement =
connection.prepareStatement(Queries.DELETE_FILLLIAL_BY_ID);
        statement.setInt(1, index);
        statement.executeUpdate();
    } catch (SQLException ex) {
        new MessageDialog(null,
true).showErrorMessage("Неудалось удалить филиал.");
    }
}
}

```

Листинг 4 – файл <Queries>

```
package database;
```

```
public interface Queries {
```

```

    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, filllial 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.`idFilllial` = f.`idFilllial`";

```

```

    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_INTO_PRODUCT = "INSERT INTO
product (mark, `idProduct`,

```

```

`idBatteryType`,`idBodyType`,`idModel`,`idPhoneType`)      VALUES
(?,?,?,?,?,?)";

    public static String INSERT_INTO_MOBILEPHONES = "INSERT INTO
mobilephone(`idMobilephone`,price,      `count`,      `idProduct`,
`idFillial`)VALUES(?,?,?,?,?)";

    public static String INSERT_INTO_PARAMETERS = "INSERT INTO
`parameters` (`idParameters`, `parameter`) VALUES (?, ?)";

    public static String INSERT_INTO_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 <AddFilllialLogic>

```
package guilogic;

import gui.AddFilllialDialog;

public class AddFilllialLogic {

    private AddFilllialDialog dialog;
    private boolean confirm;

    public AddFilllialLogic() {
        dialog = new AddFilllialDialog(null, true, this);
    }
    public void showAddFilllialDialog() {
        dialog.getjTextFieldName().setText("");
        dialog.getjTextAreaAddress().setText("");
        confirm = false;
        dialog.setVisible(true);
    }
    public void showEditFilllialDialog(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 guillogic;

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 guillogic;
```



```

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
        e())));

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

        dialog.setjComboBoxFillial(createComboBoxModel(executor.getAllFi
        llials())));

        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 filllialId = new
Scanner(dialog.getjComboBoxFilllial().getSelectedItem().toString(
)).nextInt();
            SQL.append("AND
m.`idFilllial`=").append(filllialId).append(" ");
        }
        if (dialog.getjCheckBox1()) {
            int modelId = new
Scanner(dialog.getjComboBox1().getSelectedItem().toString()).nex
tInt();
            SQL.append("AND
mo.`idModel`=").append(modelId).append(" ");
        }
        if (dialog.getjCheckBox2()) {
            int phoneTypeId = new
Scanner(dialog.getjComboBox2().getSelectedItem().toString()).nex
tInt();
            SQL.append("AND
pt.`idPhoneType`=").append(phoneTypeId).append(" ");
        }
        if (dialog.getjCheckBox3()) {
            int bodyTypeId = new
Scanner(dialog.getjComboBox3().getSelectedItem().toString()).nex
tInt();
            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 guillogic;
```

```

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 guillogic;  
  
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 guillogic;

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
        ProgressDialog(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 guillogic;

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 solded;
}
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 (solded && 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 = selled ? 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));
        }
        solded = true;
    }
    public void showSelledPhones() {
        ResultSet resultSet = executor.findAllSelledPhones();
        if (resultSet != null) {
            mobilephonesDialog.getTable1().setModel(new
TableModel(resultSet));
        }
        solded = 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;
}
}
```