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.

"____" _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

на тему:

Розроблення вебзастосунку "Адміністратор" для інформаційної системи автосалону

Виконав: студент групи 4141

_____ *Кузнецов М. М.*
(підпис)

Керівник роботи:

викладач _____
(посада, науковий ступінь вчене звання)

_____ *Бобровський О. С.*
(підпис)

м. Миколаїв – 2025 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 122 "Комп'ютерні науки"

Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"
Гарант освітньої програми
_____
к.т.н, доц. Гайда А.Ю.
"____" _____ 2025 р.

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

Студенту Кузнецов Микита Миколайович
(прізвище, ім'я, по батькові)

1. Тема роботи: Розроблення вебзастосунку "Адміністратор" для інформаційної системи автосалону

Керівник роботи Бобровський Олексій Сергійович

Затверджені наказом ректора № 295-уч від "27" березня 2025 року.

2. Термін подання роботи: 20 червня 2025 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) Розробка концепції АРМ адміністратора для продажу авто, Розробка технічного проєкту інформаційної системи АРМ, Реалізація проєкту інформаційної системи АРМ адміністратора для продажу авто, Охорона праці.

5. Перелік презентаційних матеріалів Титульний лист, Мета дослідження, Завдання досліджння, "Об'єкт, предмет та актуальність", Модель концепції, Концептуальна модель даних, Фізична модель даних, Екранні форми, Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Бобровський О. С. викладач	03.02.2025	22.02.2025
2	Бобровський О. С. викладач	22.02.2025	21.03.2025
3	Бобровський О. С. викладач	21.03.2025	25.04.2025
4	Бобровський О. С. викладач	25.04.2025	07.06.2025

7. Дата видачі завдання 03 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	13.02.2025	
2	Постановка задачі	15.02.2025	
3	Розробка концепції	22.02.2025	
4	Розробка проєкту ІС	21.03.2025	
5	Реалізація проєкту	25.04.2025	
6	Розробка питань з охорони праці	07.06.2025	
7	Розробка графічних матеріалів	12.06.2025	
8	Подання роботи на перевірку на плагіат	10.06.2025	
9	Подання роботи рецензенту	15.06.2025	
10	Подання роботи на захист	20.06.2025	

Студент _____ *Кузнецов М. М.*
(підпис) (ПІБ)

Керівник роботи _____ *Бобровський О. С.*
(підпис) (ПІБ)

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Розроблення вебзастосунку "Адміністратор" для інформаційної системи автосалону».

Актуальність теми полягає у зростаючій потребі автосалонів в оптимізації внутрішніх бізнес-процесів, централізації обліку клієнтів і замовлень, а також у забезпеченні прозорої взаємодії між адміністративними та бухгалтерськими підрозділами. В умовах цифрової трансформації бізнесу використання автоматизованих рішень стає ключовим чинником підвищення ефективності, конкурентоспроможності та рентабельності автосалонів.

Метою роботи є створення вебзастосунку для автоматизованого робочого місця адміністратора, який забезпечить централізований облік даних, інтеграцію з бухгалтерською системою, підтримку взаємодії між працівниками та підвищення якості обслуговування клієнтів.

У ході роботи було виконано аналіз предметної області, виявлено проблеми функціонування автосалону без автоматизації, сформовано технічне завдання, спроєктовано концептуальну, логічну та фізичну моделі даних, розроблено архітектуру інформаційної системи, реалізовано її функціональні компоненти, протестовано ключові модулі, а також підготовлено рекомендації щодо її впровадження та подальшої експлуатації.

Основна частина роботи містить 55 сторінок тексту, 10 рисунків, 2 таблиці, 2 додатки. Список використаних джерел налічує 20 найменувань. Робота виконана українською мовою.

Ключові слова: інформаційна система, автоматизоване робоче місце, автосалон, адміністратор, бухгалтерія, Java, Spring Boot, PostgreSQL, веб-застосунок.

ABSTRACT

Theme of the qualification work: “Development of the web application “Administrator” for the car dealership information system”.

The relevance of the topic lies in the growing need for car dealerships to optimize internal business processes, centralize customer and order accounting, and ensure transparent interaction between administrative and accounting departments. In the context of digital business transformation, the use of automated solutions is becoming a key factor in increasing the efficiency, competitiveness, and profitability of car dealerships.

The aim of the work is to create a web application for an automated administrator's workstation that will provide centralized data accounting, integration with the accounting system, support interaction between employees, and improve the quality of customer service.

In the course of the work, was analyzed the subject area, identified the problems of the car dealership functioning without automation, formed the terms of reference, designed the conceptual, logical and physical data models, developed the architecture of the information system, implemented its functional components, tested the key modules, and prepared recommendations for its implementation and further operation.

The main part of the work contains 55 pages of text, 10 figures, 2 tables, 2 appendices. The list of references includes 20 items.

The paper is written in Ukrainian.

Keywords: information system, automated workplace, car dealership, administrator, accounting, Java, Spring Boot, PostgreSQL, web application.

ЗМІСТ

Перелік умовних позначень та скорочень 7
Вступ 8
Розділ 1 Розробка концепції АРМ адміністратора для продажу авто 10
1.1 Опис об'єкту дослідження і аналіз предметної області 10
1.2 Формування вимог до АРМ адміністратора 14
1.3 Розробка концепції АРМ адміністратора 17
Розділ 2 Розробка технічного проєкту інформаційноЇ системи АРМ адміністратора для продажу авто 21
2.1 Загальносистемні рішення 21
2.2 Рішення з інформаційного забезпечення 24
2.2.1 Розробка концептуальної моделі даних 25
2.2.2 Розробка логічної моделі бази даних 29
2.2.3 Розробка фізичної моделі бази даних 30
2.3 Рішення з програмного забезпечення 31
2.4 Рішення з технічного забезпечення 32
Розділ 3 Реалізація проєкту інформаційної системи АРМ адміністратора для продажу авто 34
Розділ 4 Охорона праці 40
Висновок 43
Список використаної літератури 45
Додаток А 47
Додаток Б 49

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

АРМ - Автоматизоване робоче місце.

ІС - Інформаційна система.

БД - База даних.

СКБД - Система керування базами даних.

ER - Сутність-зв'язок.

SQL - Мова структурованих запитів.

API - Інтерфейс прикладного програмування.

ВСТУП

У наш час, коли бізнес усе активніше переходить у цифровий формат, інформаційні системи стають важливим інструментом для кращого управління підприємствами. Вони допомагають швидше виконувати повсякденні завдання, зберігати дані, покращувати обслуговування клієнтів і збільшувати прибутки.

Автомобільний ринок України є одним із найбільш динамічних, і конкуренція в цій сфері стимулює підприємства до впровадження цифрових рішень, які дозволяють ефективно керувати процесами продажу, логістики, фінансів та обслуговування після продажу. Особливого значення набуває автоматизація адміністративної частини автосалону, яка охоплює обробку замовлень, облік клієнтів, управління персоналом та взаємодію з бухгалтерськими службами.

Актуальність теми полягає в необхідності впровадження сучасного вебзастосунку, який забезпечить централізоване управління адміністративними процесами автосалону, мінімізує дублювання інформації, зменшить ризик помилок і втрат, а також дозволить забезпечити інтеграцію з бухгалтерськими системами.

Об’єктом дослідження є процес адміністрування інформаційних потоків у діяльності автосалону.

Предметом дослідження є принципи, методи та інструменти розробки інформаційної системи автоматизованого робочого місця адміністратора, орієнтованої на централізоване управління обліковими та фінансовими операціями.

У прикладних розробках і публікаціях останніх років активно розглядаються питання цифровізації бізнесу, автоматизації облікових процесів і побудови інформаційних систем у сфері торгівлі. Проблематика інтеграції з бухгалтерськими рішеннями, ефективного обміну даними та оптимізації бізнес-

процесів є актуальною для автосалонів, які прагнуть підвищити конкурентоспроможність.

Мета кваліфікаційної роботи — розробити вебзастосунок “Адміністратор” для інформаційної системи автосалону, що забезпечить автоматизацію обліку замовлень, клієнтів, фінансових транзакцій і адміністративної взаємодії персоналу.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

1. Проаналізувати предметну область діяльності автосалону, виявити недоліки чинних підходів до адміністрування;

2. Сформулювати вимоги до інформаційної системи та обґрунтувати вибір засобів її реалізації;

3. Розробити концептуальну, логічну та фізичну моделі бази даних;

4. Побудувати програмну архітектуру системи, реалізувати вебінтерфейс адміністратора та забезпечити обробку замовлень і клієнтів;

5. Інтегрувати систему з типовим бухгалтерським рішенням та реалізувати механізми обміну платіжною інформацією;

6. Провести тестування функціоналу та сформувати висновки щодо ефективності запропонованого рішення;

7. Розглянути вимоги з охорони праці при організації робочого місця адміністратора.

Таким чином, реалізація зазначених завдань дозволяє створити ефективну, зручну та безпечну інформаційну систему, яка відповідає актуальним вимогам ринку та сприяє підвищенню продуктивності праці в автосалоні.

РОЗДІЛ 1 РОЗРОБКА КОНЦЕПЦІЇ АРМ АДМІНІСТРАТОРА ДЛЯ ПРОДАЖУ АВТО

1.1 Опис об'єкту дослідження і аналіз предметної області

Автомобільний ринок України залишається одним із найбільш динамічних і конкурентних сегментів економіки. У 2023 році ринок нових легкових автомобілів виріс на понад 49% порівняно з попереднім роком. Основними чинниками такого зростання стали стабілізація економіки, активне повернення громадян з-за кордону, а також розвиток програм лізингу та кредитування. Водночас очікування клієнтів істотно змінилися: покупці потребують швидкого, персоналізованого сервісу, доступного онлайн і без зайвих черг або паперової тяганини.

Автосалон — це багатофункціональне підприємство, діяльність якого включає продаж автомобілів, супровід клієнтів, оформлення фінансових операцій, організацію технічного обслуговування та підтримку після продажу. Його функціонування базується на чіткій взаємодії кількох ключових ролей, кожна з яких виконує власний набір завдань у загальному бізнес-процесі.

Структура автосалону наведена на рисунку 1.1:

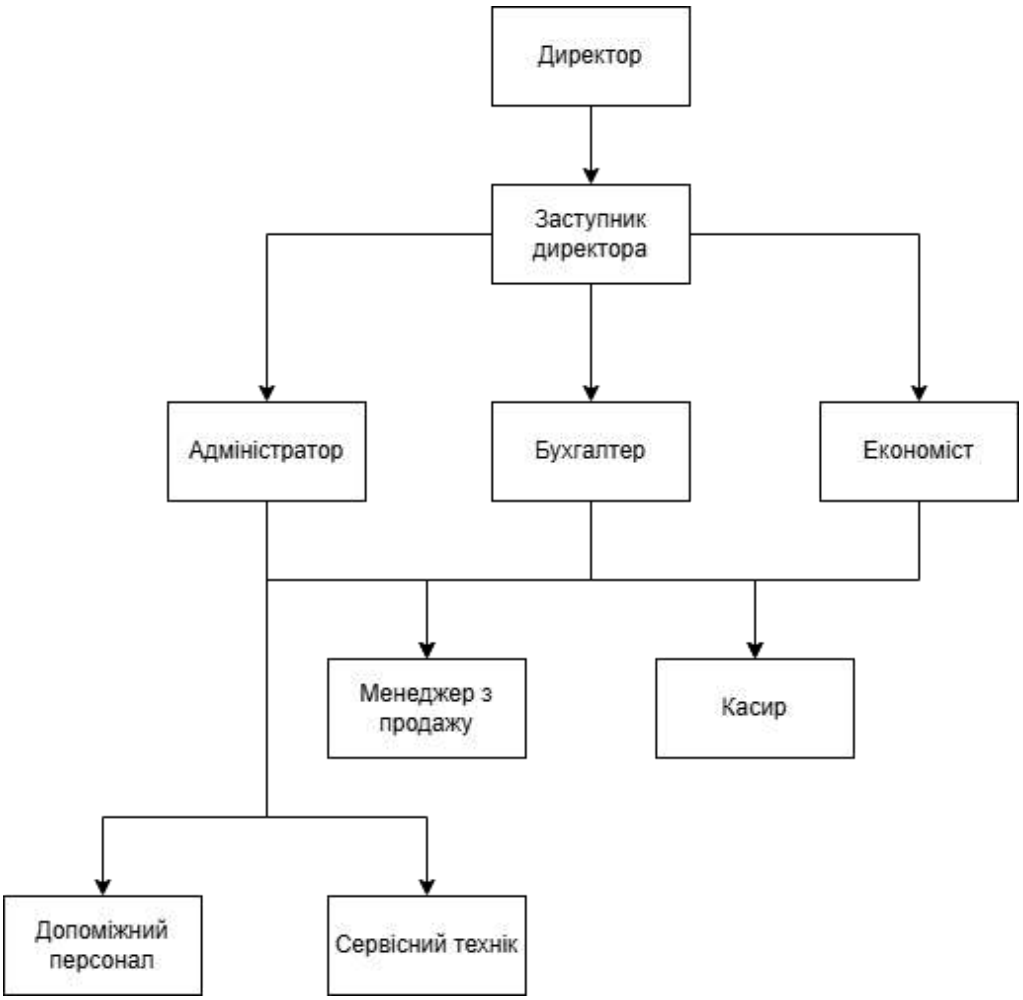

Рисунок 1.1 - Організаційна структура автосалону

До основних ролей, що забезпечують функціонування автосалону, належать:

- Директор — керує загальною стратегією розвитку, ухвалює фінансові рішення, контролює діяльність персоналу.

- Заступник директора — допомагає координувати підрозділи, виконує нагляд за поточними операціями.

- Менеджери з продажу — займаються консультуванням клієнтів, презентацією авто, підбором відповідних моделей, оформленням замовлень.

- Адміністратор — координує внутрішні процеси, веде облік клієнтів, замовлень і персоналу, формує управлінську звітність, забезпечує актуальність даних.

- Бухгалтер — відповідає за виставлення рахунків, контроль оплат, фінансовий облік і складання звітності.

- Економіст — планує закупівлі, облік залишків, оптимізує логістику.

- Касир — приймає платежі через готівку та термінали.

- Фахівець із технічного обслуговування — готує автомобілі до продажу, проводить діагностику та сервісні роботи.

- Обслуговуючий персонал — забезпечує чистоту території, адміністративне обслуговування [1].

Незважаючи на центральну роль адміністратора в управлінні внутрішніми процесами автосалону, його діяльність стикається з низкою суттєвих обмежень через відсутність належної автоматизації. Адміністратор відповідає за ведення обліку клієнтів, замовлень, персоналу, а також за формування звітності та координацію дій між відділами. Проте в поточних умовах ця діяльність ускладнена з таких причин [2]:

1. Документообіг ведеться вручну або в різнорідних локальних програмах. Адміністратори змушені вести записи у звичайних таблицях або блокнотах, інколи використовуються файли Excel чи інші офісні програми без зв’язку між ними. Це значно ускладнює пошук та оновлення інформації, створює ризики дублювання та втрати даних.

2. Відсутня централізована база даних. Вся інформація про клієнтів, замовлення, оплату тощо зберігається у вигляді фрагментів. Адміністратор не має оперативного доступу до повної історії взаємодії з клієнтом або до стану виконання замовлення. Це гальмує обслуговування та заважає ухваленню своєчасних рішень.

3. Адміністратор не має засобів контролю за діями інших учасників процесу. У разі потреби перевірити, чи відправлено автомобіль клієнту, або чи була проведена оплата, адміністратор змушений звертатися до інших співробітників особисто, що затягує виконання завдань.

4. Немає єдиного підходу до обліку. Оскільки кожен підрозділ використовує власні інструменти або шаблони для обробки інформації, адміністратор не може узгодити ці дані автоматично, що створює плутанину в документації.

5. Обмін даними здійснюється вручну. Узгодження замовлень, оплат або логістичних операцій відбувається через особисті повідомлення, паперові бланки або телефонні дзвінки. Це не лише знижує швидкість обробки інформації, а й підвищує ризик помилок та втрати важливої інформації.

6. Адміністратор не має інструментів аналітики. У зв'язку з відсутністю автоматизованої системи адміністратор не може швидко сформувати звіти про кількість замовлень, клієнтів або стан виконання договорів. Це унеможливлює ефективне управління ресурсами.

Відсутність централізованого IT-рішення створює бар’єри в комунікації між працівниками, ускладнює облік та аналітику, що прямо впливає на рентабельність і здатність швидко адаптуватися до потреб клієнтів.

Через ці чинники наявна низька рентабельність бізнесу, виникають додаткові витрати, збільшується час обслуговування клієнтів, втрачається конкурентна перевага, бухгалтерський облік та аналітика ускладнені, немає можливості якісного навчання нових кадрів.

З метою вирішення виявлених проблем передбачається розробка автоматизованого робочого місця адміністратора із можливістю централізованого обліку клієнтів, замовлень, платежів, взаємодії з бухгалтерією та аналітики бізнес-процесів. Інтегрування готового типового рішення для бухгалтерського обліку, дозволить забезпечити надійність і повноту фінансового контролю без потреби в створенні власного бухгалтерського модуля з нуля. Це дозволить збільшити швидкість навчання та адаптації нових співробітників, покращить масштабованість та дозволить використати створену системи для відкриття нових, або оптимізацію інших автосалонів компанії [1].

Результатом має стати покращення рентабельності, оптимізація всіх адміністративних і фінансових процесів, підвищення ефективності роботи, поліпшення взаємодії персоналу, спрощення найму та навчання нових кадрів та зниження витрат на операційну діяльність.

1.2 Формування вимог до АРМ адміністратора

Для реалізації функціоналу автоматизованого робочого місця адміністратора передбачається створення вебсайту автосалону, який стане центральною платформою для взаємодії користувачів із системою. Через нього користувачі зможуть:

- переглядати інформацію про наявні автомобілі;
- надсилати заявки та отримувати зворотній зв’язок;
- взаємодіяти з менеджером для консультацій;
- отримувати доступ до акційних пропозицій або знижок.

Адміністратор системи працюватиме через спеціалізовану веб-панель, інтегровану в сайт, яка забезпечить доступ до всіх необхідних функцій керування замовленнями, клієнтами, персоналом, звітністю та аналітикою. Бухгалтер, у свою чергу, матиме доступ до окремого захищеного інтерфейсу, синхронізованого з типовим програмним рішенням для бухгалтерського обліку, що дозволить здійснювати облік транзакцій, формування звітності та фінансовий контроль.

Таким чином, вебсайт стає центральним елементом архітектури системи, який поєднує користувача, адміністратора, бухгалтера, базу даних та зовнішні сервіси (API банку, поштовий сервер тощо) через клієнт-серверну модель [3].

Автоматизована інформаційна система автосалону має забезпечувати виконання таких основних функціональних завдань, необхідних для ефективної роботи адміністратора:

- централізований облік клієнтів, автомобілів та замовлень;
- ведення і контроль статусів замовлень;

- формування звітів про виконані операції та стан продажів;

- надання можливості коментування та узгодження дій між адміністратором та бухгалтером;

- підтримка обміну інформацією з бухгалтерською системою;

- організація роботи з оплатами та контроль фінансових транзакцій;

- забезпечення збереження історії змін і аудиту дій користувачів;

- можливість масштабування системи в залежності від зростання кількості даних і користувачів;

- підтримка роботи як у локальному середовищі, так і в хмарних інфраструктурах.

Крім того, система повинна мати такі нефункціональні характеристики:

- наявність стабільного підключення до Інтернету для інтеграції з зовнішніми сервісами;

- зручний, інтуїтивно зрозумілий графічний інтерфейс адміністратора;

- підтримка адаптивного дизайну для різних типів пристроїв;

- можливість простого розгортання та міграції системи між різними середовищами;

Для забезпечення надійної та ефективної роботи АРМ адміністратора необхідно обрати відповідне програмне забезпечення, яке охоплює інструменти для розробки, налагодження, запуску системи та її подальшої підтримки. Вимоги до програмних ресурсів включають такі компоненти [3]:

Мова програмування: Java 17. Вибір зумовлений її стабільністю, широкою підтримкою в корпоративних системах та сумісністю з технологією Spring Boot.

Фреймворк: Spring Boot. Забезпечує швидку розробку веб-застосунків, має вбудовану підтримку REST API, інтеграцію з базами даних, безпекою та модульністю.

Система керування базами даних: PostgreSQL. Це надійна реляційна СКБД з високим рівнем масштабованості, підтримкою транзакцій, перевіреним механізмом збереження цілісності даних та активною спільнотою [8].

Середовище розробки: IntelliJ IDEA. Забезпечує розширену підтримку Java-проєктів, має інтеграцію з Maven, Git, Spring Boot, а також зручний інтерфейс для відлагодження [9].

Система керування проєктами та залежностями: Maven. Дає змогу централізовано керувати бібліотеками, збіркою і структурами модулів.

Операційна система: Ubuntu Server 22.04 LTS. Вибір обґрунтований стабільністю, довгостроковою підтримкою та поширеністю серед серверних рішень.

Для забезпечення розгортання, обслуговування та підтримки системи застосовуються також допоміжні програмні засоби:

Postman — для тестування REST API запитів і перевірки правильності взаємодії між клієнтською та серверною частиною.

pgAdmin — графічна утиліта для адміністрування бази даних PostgreSQL, яка дозволяє переглядати, змінювати та аналізувати структуру даних.

Git та GitHub — для контролю версій і спільної роботи над кодом. Дозволяють ефективно відстежувати зміни, створювати резервні копії проєкту, реалізовувати командну розробку.

TeamViewer — інструменти для віддаленого доступу до серверів чи клієнтських машин, необхідні для адміністрування системи в разі технічних проблем або налаштування.

Засоби захисту: використання HTTPS, обмеження доступу через фаєрволи, автентифікація користувачів за ролями. Це забезпечує конфіденційність та захист даних від несанкціонованого доступу.

Поштова служба Gmail SMTP — Використовується для надсилання повідомлень клієнтам, нагадувань та службової кореспонденції.

Google Chrome — для доступу до веб-інтерфейсу системи з боку користувача.

Google Cloud: використовується для хостингу серверної частини та забезпечення масштабованості.

Вимоги до апаратного забезпечення серверу:

Процесор: не нижче AMD Ryzen 5 5600 або аналогічний, з мінімум 6 ядрами (12 потоків);

Оперативна пам’ять: не менше 16 ГБ DDR4;

Накопичувач: NVMe SSD об’ємом від 1 ТБ;

Мережева інфраструктура: Гігабітний Ethernet (1 Gbps);

Операційна система: стабільний Linux-дистрибутив (наприклад, Ubuntu Server 22.04 LTS);

Джерело безперебійного живлення (UPS).

Таким чином, на основі сформованих вимог розробляється концепція АРМ адміністратора.

1.3 Розробка концепції АРМ адміністратора

Автоматизоване робоче місце адміністратора є ключовим компонентом інформаційної системи автосалону, яке забезпечує централізоване управління основними бізнес-процесами та даними. Концепція АРМ адміністратора передбачає формування інтегрованого середовища для обробки, контролю та аналізу інформації про автомобілі, клієнтів, співробітників, угоди, замовлення та платежі. На рисунку 1.3 наведена розроблена модель концепції інформаційної системи:

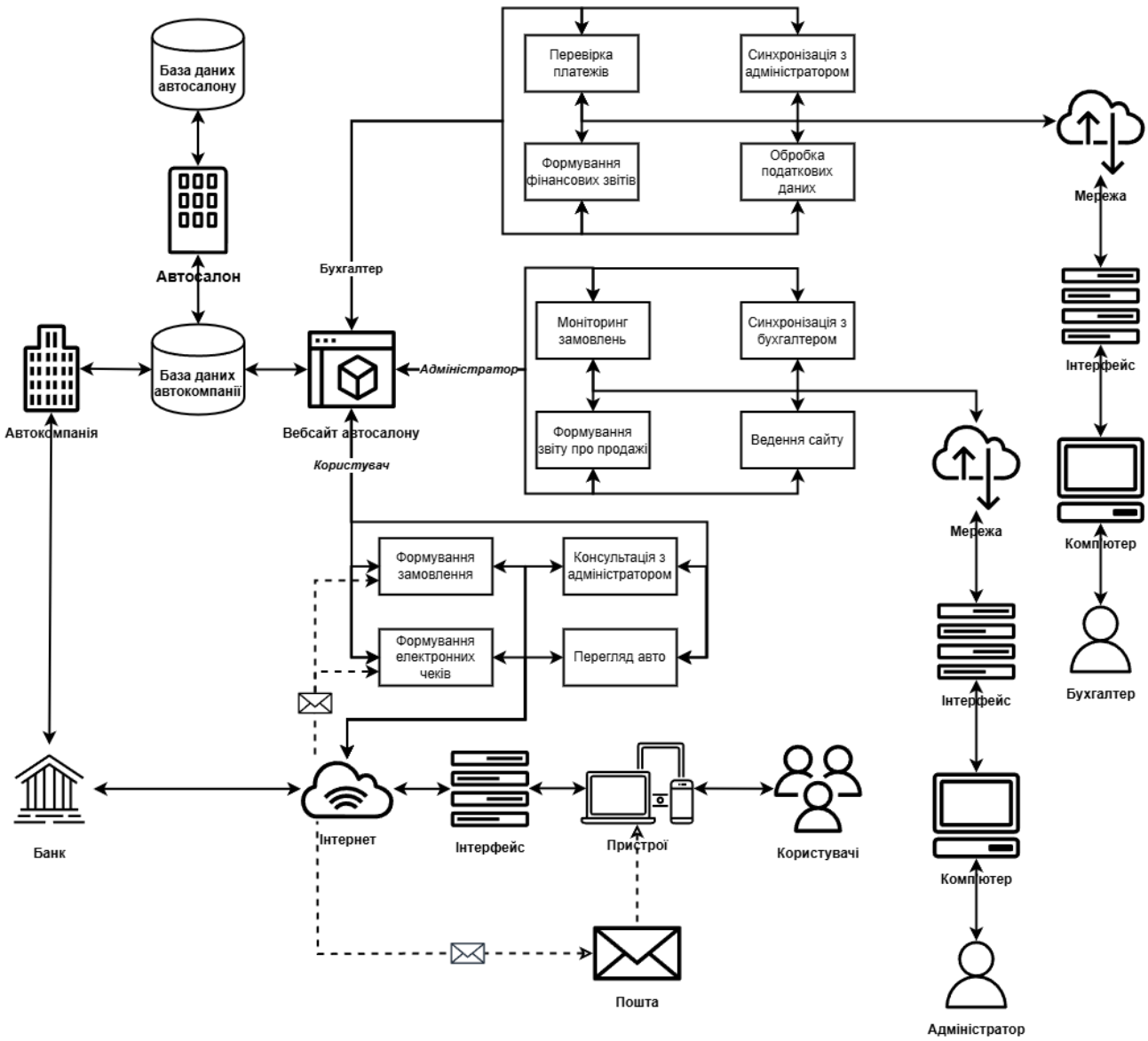

Рисунок 1.3 - Модель концепції

Основним завданням АРМ є підтримка повного циклу операцій з даними, включно зі створенням, переглядом, редагуванням і видаленням записів. Крім того, передбачена можливість моніторингу статусу замовлень і угод, формування звітності та ведення обліку взаємодії з клієнтами [2].

Особливу увагу приділено механізмам розмежування доступу, що забезпечують безпеку даних та відповідність посадовим обов’язкам. Зокрема, реалізується рольова модель із різними рівнями прав, яка дозволяє адміністраторам виконувати повний контроль за роботою системи, бухгалтерам

— опрацьовувати фінансові документи, а менеджерам — керувати клієнтськими запитами.

Для забезпечення ефективної роботи АРМ передбачено інтеграцію з зовнішніми сервісами, зокрема: електронною поштою для оперативного обміну повідомленнями, банківськими системами для автоматизації платіжних операцій, а також рекламними платформами для підтримки маркетингової діяльності автосалону.

З огляду на сучасні вимоги, система розробляється за клієнт-серверною архітектурою з веб-інтерфейсом, що гарантує зручний доступ через будь-який браузер і підтримку різних пристроїв (десктопи, планшети). Для роботи адміністратора передбачено стандартне робоче місце з ПК, оснащене необхідним програмним забезпеченням і стабільним доступом до інтернету.

Нефункціональні вимоги концепції включають забезпечення стабільної роботи системи навіть при високих навантаженнях, масштабованість, безпеку даних і стійкість до зовнішніх загроз. Для цього планується використання сучасних технологічних підходів, що дозволять ефективно обробляти великі обсяги інформації та підтримувати цілісність системи.

Вимоги до апаратного забезпечення робочого місця адміністратора включають сучасний багатоядерний процесор, не менше 16 ГБ оперативної пам’яті, швидкий твердотільний накопичувач і надійне інтернет-з’єднання, що забезпечать безперебійну роботу з системою.

Таким чином, розроблена концепція АРМ адміністратора формує основу для побудови ефективної, безпечної і масштабованої інформаційної системи автосалону, яка забезпечить цифрову трансформацію бізнес-процесів і підвищення продуктивності праці персоналу.

Висновок до розділу 1:

Було проведено аналіз предметної області автосалону, виявлено основні проблеми, пов’язані з відсутністю централізованої інформаційної системи.

Сформульовано функціональні та нефункціональні вимоги до автоматизованого робочого місця адміністратора, розроблено концепцію майбутньої системи, яка враховує потреби адміністраторів, бухгалтерів та клієнтів. Запропоновано архітектуру, що дозволяє ефективно автоматизувати бізнес-процеси та підвищити якість обслуговування.

РОЗДІЛ 2 РОЗРОБКА ТЕХНІЧНОГО ПРОЄКТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ АРМ АДМІНІСТРАТОРА ДЛЯ ПРОДАЖУ АВТО

2.1 Загальносистемні рішення

Для побудови ефективної інформаційної системи автосалону важливим є визначення загальносистемних рішень, які стосуються вибору апаратної архітектури, платформи розробки, бази даних, програмного стеку, а також засобів інтеграції з зовнішніми сервісами. Одним із ключових аспектів є вибір типового програмного рішення для ведення бухгалтерського обліку, податкової та фінансової звітності. Сучасні автосалони потребують надійних, автоматизованих рішень, які дозволяють контролювати фінансові потоки, працювати з касою, формувати звітність для податкових органів, обробляти розрахунки з клієнтами та постачальниками.

Нижче представлено порівняльний аналіз трьох найпоширеніших ТПР в Україні, які можуть бути використані в автосалонах наведено нижче у таблиці 2.1:

Таблиця 2.1 - Порівняння характеристик ТПР для бухгалтерського обліку

Критерій	BAS Бухгалтерія	SAP Business One	QuickBooks Online
Юридична підтримка в Україні	Повна	Обмежена	Відсутня
Мова інтерфейсу	Українська	Англійська	Англійська
Хмарна версія	Так	Так	Так

Автоматизація податкової звітності	Повна	Обмежена	Відсутня
Інтеграція з банками та касовими апаратами	Так	Обмежено	Ні
Масштабованість	Середня	Висока	Низька
Вартість впровадження	Помірна	Дуже висока	Низька
Відгуки користувачів	Позитивні	Складне налаштування	Обмежена функціональність
Підтримка обліку ТМЦ та послуг	Так	Так	Частково

BAS Бухгалтерія є найбільш адаптованим продуктом для українського ринку. Рішення підтримується українським розробником та постійно оновлюється відповідно до змін у законодавстві. BAS дозволяє формувати електронну звітність, працювати з податковими накладними, обліковувати основні засоби, товари, послуги, вести зарплатні відомості та автоматизувати більшість бізнес-процесів, що відповідає вимогам середнього автосалону.

SAP Business One — потужна ERP-система з широкими можливостями для корпоративного сектору. Водночас вона вимагає значних фінансових та часових

ресурсів для впровадження й обслуговування, що не є оптимальним для середнього автосалону.

QuickBooks Online — зручний хмарний сервіс, популярний у США, однак не має адаптації під українське законодавство та інтеграцій з українськими банками.

З огляду на вищенаведене, оптимальним вибором ТПР для бухгалтерського та фінансового обліку в АРМ адміністратора автосалону є BAS Бухгалтерія. Вона повністю відповідає вимогам українського ринку, забезпечує необхідну інтеграцію з банками, ПРРО, платіжними системами, а також підтримує автоматизацію обліку продажів, складу, постачань та фінансів [1].

На основі обраного рішення передбачається інтеграція системи управління автосалоном з BAS Бухгалтерія через обмін документами у форматах XML або JSON, з можливістю автоматичного формування документів (рахунків, видаткових накладних, актів приймання-передачі) для подальшого перенесення в BAS.

Також у складі загальносистемних рішень були визначені:

- Архітектура: багаторівнева клієнт-серверна з чітким розподілом обов’язків;

- База даних: PostgreSQL як потужна об’єктно-реляційна СКБД;

- Стек технологій: Java + Spring Boot + Hibernate + HTML/CSS/JS;

- Платформа розробки: IntelliJ IDEA для backend, Postman для API-тестування, GitHub для спільної роботи над кодом;

- Система захисту даних: аутентифікація, авторизація, шифрування переданих даних, обмеження доступу за ролями;

- Можливість масштабування: горизонтальна та вертикальна для майбутнього зростання системи.

Загальносистемні рішення заклали основу для побудови ефективної, безпечної та масштабованої інформаційної системи для АРМ адміністратора автосалону.

2.2 Рішення з інформаційного забезпечення

Інформаційне забезпечення є одним із ключових компонентів будь-якої інформаційної системи. Воно охоплює всі структури, механізми та моделі, які використовуються для зберігання, обробки та організації даних, що формують інформаційне ядро проєкту. У рамках розробки автоматизованого робочого місця адміністратора для автосалону інформаційне забезпечення реалізується за допомогою реляційної бази даних, яка забезпечує централізоване, надійне та безпечне зберігання всієї необхідної інформації [].

У даному проєкті для зберігання даних було обрано сучасну об’єктно-реляційну систему управління базами даних PostgreSQL. Це потужна, масштабована та відкрита СКБД, яка підтримує стандарти SQL, транзакції, зв’язки між таблицями, індексацію, тригери, збережені процедури та багато інших можливостей, необхідних для побудови складних інформаційних систем.

PostgreSQL вирізняється серед інших СКБД такими характеристиками [8]:

- Надійність і стійкість: завдяки серйозному підходу з використанням транзакцій (ACID-сумісність) ми маємо чудово цілісність даних навіть у разі помилок або збоїв;
- Масштабованість: підтримує роботу з великими обсягами даних та розширення структури таблиць без значного впливу на продуктивність;
- Гнучкість: дозволяє створювати складні запити, представлення, індекси для підвищення швидкості доступу до даних;
- Підтримка ORM: відмінно інтегрується з Java-технологіями через фреймворк Hibernate і Spring Data JPA, що значно полегшує розробку;
- Безпека: підтримка контролю доступу, шифрування, сертифікатів SSL, аудиту дій користувачів;
- Відкритий код: PostgreSQL є повністю безкоштовною і має активну спільноту розробників, що забезпечує швидке усунення вразливостей та розвиток функціоналу.

Для керування базою даних у процесі розробки використовувався графічний інструмент pgAdmin, що дозволяє зручно створювати, редагувати та візуалізувати структури бази, а також виконувати SQL-запити, перевіряти індекси, перевіряти продуктивність та здійснювати резервне копіювання.

Таким чином, PostgreSQL було обрано як оптимальне рішення для даного проєкту, що забезпечує надійне функціонування інформаційної системи, простоту інтеграції з Java-застосунком і гнучке масштабування у разі зростання навантаження.

Для формалізації структури даних, що зберігаються в базі, та забезпечення логічного узгодження між усіма компонентами системи, наступним кроком є розробка концептуальної моделі даних.

2.2.1 Розробка концептуальної моделі даних

Концептуальна модель даних описує предметну область на абстрактному рівні, незалежно від конкретних реалізацій. Основна мета — зафіксувати головні сутності, атрибути та зв’язки, які відображають реальні об’єкти та процеси, що відбуваються в діяльності автосалону. Такий підхід дозволяє краще структурувати інформацію та забезпечити цілісність у подальших етапах моделювання [7].

На етапі формування концептуальної моделі було виділено ключові сутності: клієнт, автомобіль, менеджер, замовлення, додаткові опції, типи кузова, коробки передач, приводу, палива тощо. Їх атрибути та типи ключів наведено у таблиці 2.2.

Таблиця 2.2 - Сутності, атрибути та ключі концептуальної моделі даних

Сутність	Атрибути	Тип ключу
Клієнт	ID	Primary
	Ім'я та прізвище	

	Адреса	
	Електронна пошта	
	Номер мобільного телефону	
	Номер паспорту	
	Ідентифікаційний код	
	Номер посвідчення водія	
	Пароль	
	Повноваження	
Заявка	ID	Primary
	Дата створення	
	Дата виконання	
	Номер замовлення	Foreign
	Номер стану замовлення	Foreign
	Номер авто	Foreign
	Номер менеджеру	Foreign
Менеджер	ID	Primary
	Ім'я та прізвище	
	Адреса	
	Електронна пошта	
	Номер мобільного телефону	
	Номер паспорту	
	Пароль	
	Повноваження	
Авто	ID	Primary
	Модель	
	Рік випуску	
	Ціна	
	ID типу кузова	Foreign

	ID типу коробки перемкнення передач	Foreign
	ID типу приводу	Foreign
	Кількість дверей	
	Місткість	
	Габаритні розміри	
	Колісна база	
	Максимальний крутний момент двигуна	
	Об’єм двигуна	
	Потужність	
	Вага	
	Кліренс	
	Вантажопідйомність	
	Об’єм багажника	
	Колір	
	Зображення	
Замовлення	ID	Primary
	Номер авто	Foreign
	Номер клієнту	Foreign
	Номер менеджеру	Foreign
	Ціна усього замовлення	
	Дата	
Додаткові опції	ID	Primary
	Назва	
	Ціна опції	
Тип кузова	ID	Primary
	Назва	
Тип палива	ID	Primary
	Назва	

Тип коробки перемкнення передач	ID	Primary
	Назва	
Тип приводу	ID	Primary
	Назва	

Зв’язки між об'єктами моделюють характер взаємодії — наприклад, замовлення створює клієнт, автомобіль може мати декілька додаткових опцій, а менеджер опрацьовує заявки. Процес розробки концептуальної моделі даних базувався на аналізі функціональних вимог до системи, з урахуванням типових рішень для автоматизованих систем управління підприємствами [4].

Встановлені зв’язки між сутностями були відображені на діаграмі концептуальної моделі даних, поданій на рисунку 2.1:

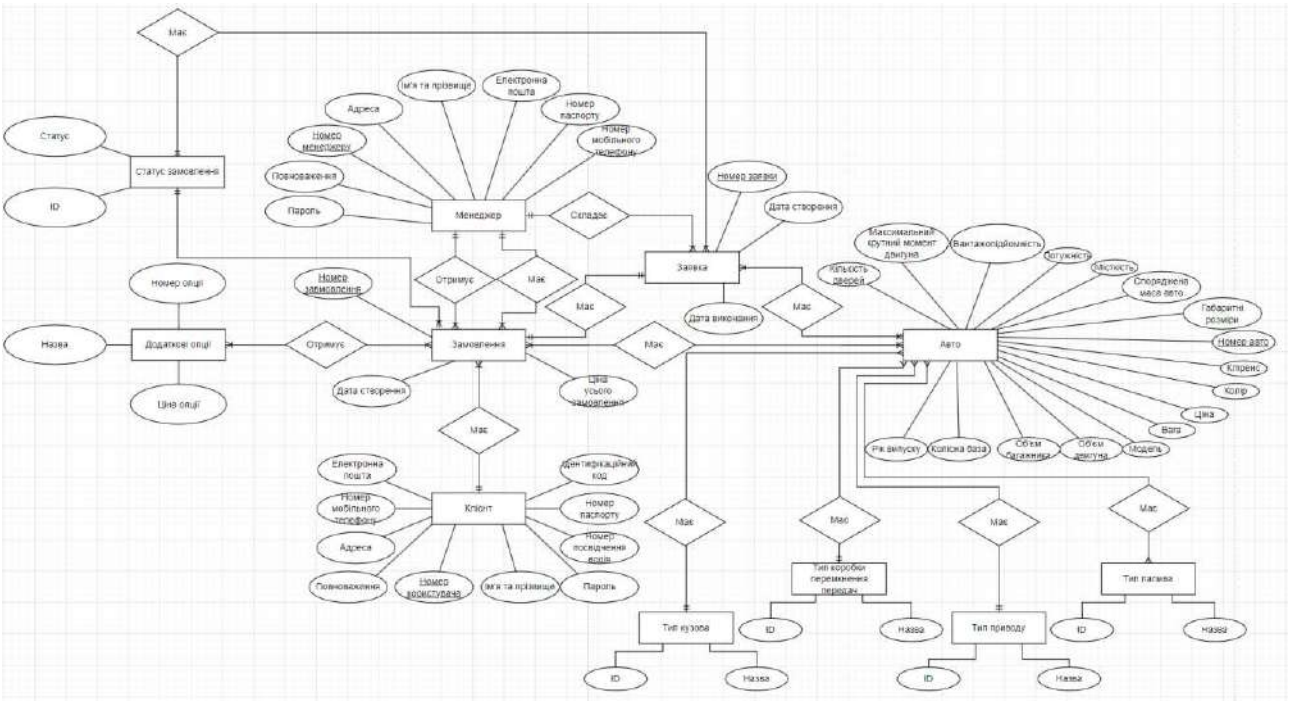

Рисунок 2.1 - Концептуальна модель даних

2.2.2 Розробка логічної моделі бази даних

Логічна модель даних є результатом деталізації концептуальної моделі з урахуванням специфіки вибраної СКБД — у нашому випадку, PostgreSQL. Вона використовується для структурування інформації у вигляді таблиць, встановлення логічних зв’язків між ними, визначення первинних та зовнішніх ключів, що забезпечує логічну цілісність даних і дозволяє уникнути надлишковості.

Логічна модель відображає структуру та взаємозв’язки реквізитного складу бази даних. Вона була розроблена на основі концептуальної моделі та адаптована під технічні вимоги системи. Призначення логічної моделі — слугувати основою для фізичної реалізації бази даних, а також для візуального представлення структури системи у вигляді схем, зрозумілих як розробникам, так і аналітикам [8].

На рисунку 2.2 зображено логічну модель бази даних, що використовується в системі:

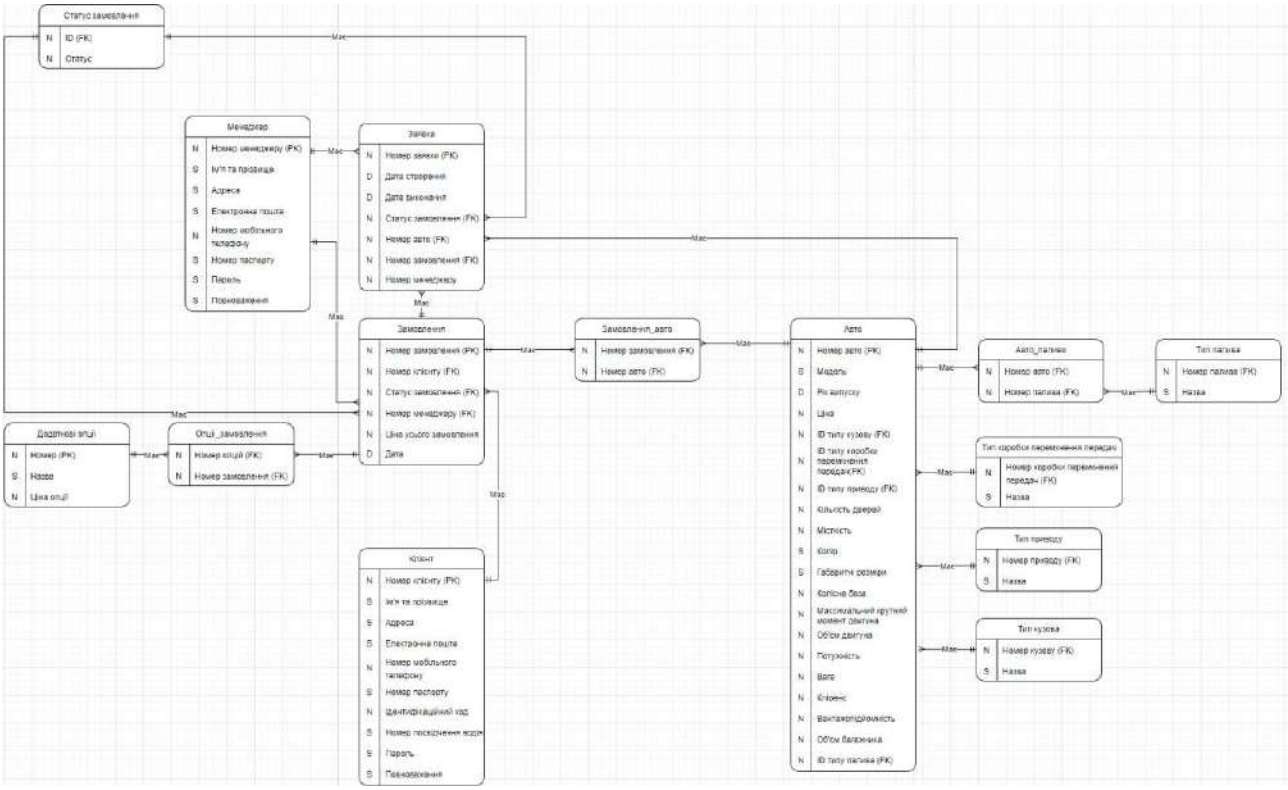

Рисунок 2.2 - Логічна модель даних

2.2.3 Розробка фізичної моделі бази даних

Фізична модель даних — це модель, яка описує конкретну реалізацію бази даних у вибраній СКБД, із зазначенням усіх технічних деталей: назв таблиць, типів даних полів, індексів, обмежень, правил зв’язку, каскадних операцій та ін. Цей рівень моделювання є основою для написання SQL-скриптів та створення структури бази даних безпосередньо в системі PostgreSQL [7].

Метою побудови фізичної моделі є створення реального середовища для зберігання та обробки даних, що забезпечує ефективне виконання запитів, підтримку транзакцій, захист цілісності даних та масштабованість. Фізична модель необхідна насамперед для розробників і адміністраторів баз даних, які реалізують систему та відповідають за її технічну підтримку.

На рисунку 2.3 наведено фізичну модель бази даних, яка використовується в системі:

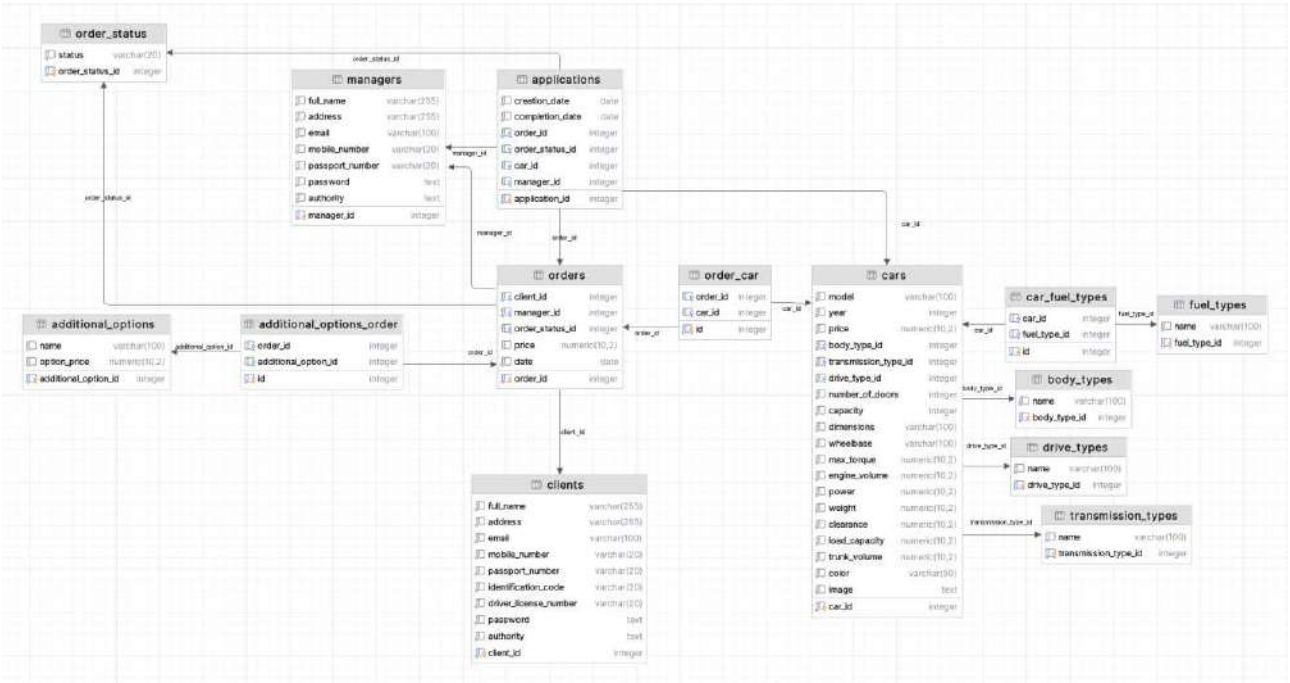

Рисунок 2.3 - Фізична модель даних

2.3 Рішення з програмного забезпечення

Програмне забезпечення автоматизованого робочого місця адміністратора автосалону реалізовано у вигляді веборієнтованого клієнт-серверного застосунку. Основною метою є забезпечення зручної, надійної та масштабованої системи керування даними про клієнтів, працівників, автомобілі, продажі, постачальників, замовлення та оплату.

Розробка системи здійснювалася з використанням таких технологій:

- Мова програмування: Java;
- Фреймворк: Spring Boot;
- Система управління базами даних: PostgreSQL;
- Інструменти ORM: Hibernate;
- Інтерфейсна частина: HTML, CSS, JavaScript;
- Інструменти розробки: IntelliJ IDEA, Postman, pgAdmin;
- Система керування версіями: GitHub.

Серверна частина реалізує REST API для взаємодії з клієнтською частиною та базою даних, забезпечує обробку запитів, валідацію даних, обробку бізнес-логіки та забезпечення безпеки [10].

Особливу увагу приділено модулю роботи з оплатами, що є критичним компонентом для системи продажу автомобілів. Підтримуються такі форми розрахунку:

- готівковий платіж;
- оплата банківською карткою через POS-термінал;
- безготівковий банківський переказ;
- часткова передоплата (у разі замовлення автомобіля);
- оплата онлайн за допомогою платіжної системи.

У системі передбачено інтеграцію з еквайринг-системою ПриватБанку через сервіс LiqPay. Така інтеграція дозволяє здійснювати:

- прийом платежів з карток Visa та MasterCard;
- оплату через Apple Pay та Google Pay;

- генерацію платіжних посилань;
- отримання статусів транзакцій у реальному часі;
- формування фінансової звітності на основі даних про проведені операції.

Використання LiqPay API є доцільним для проєкту з огляду на:

- швидке впровадження та детальну документацію;
- високий рівень безпеки, відповідність стандартам PCI DSS;
- підтримку сучасних платіжних методів;
- стабільну роботу та супровід з боку ПриватБанку.

Інтеграція з платіжною системою реалізується шляхом надсилання запитів до REST API LiqPay з боку backend-програми, з використанням ключів авторизації, які генеруються в особистому кабінеті мерчанта. Після обробки платежу система отримує callback з підтвердженням транзакції та оновлює відповідну інформацію в базі даних [9].

Таким чином, програмне забезпечення АРМ адміністратора забезпечує не лише ефективну роботу з базами даних та інформаційними потоками, а й підтримку повного циклу комерційної діяльності, включно з прийомом оплат та інтеграцією з фінансовими сервісами. Це дозволяє створити цілісну інформаційну систему, що покриває ключові потреби автосалону.

2.4 Рішення з технічного забезпечення

Технічне забезпечення охоплює апаратні засоби, необхідні для функціонування інформаційної системи, а також хмарну або серверну інфраструктуру. У проєкті передбачено використання:

- Серверної частини (може бути як локальний сервер, так і хмарна платформа типу Heroku, Railway чи AWS для хостингу back-end частини);
- Клієнтських пристроїв — комп’ютери або планшети для доступу до вебінтерфейсу адміністратора;
- Бази даних — PostgreSQL, що може розгортатися як локально, так і в хмарі (наприклад, ElephantSQL, Supabase);

- Підключення до платіжних терміналів або сканерів QR-кодів — для здійснення платежів через LiqPay/ПриватБанк;
- Резервне копіювання — регулярні бекапи БД для забезпечення цілісності та відновлення даних у разі аварій.

Ці технічні рішення забезпечують стабільну роботу інформаційної системи, її захист, надійність та зручність для кінцевих користувачів.

Висновок до розділу 2:

Було визначено загальносистемні, інформаційні, програмні та технічні рішення для реалізації інформаційної системи. Обґрунтовано вибір програмного забезпечення та інструментів, побудовано концептуальну, логічну та фізичну моделі бази даних. Обрані технології та архітектура забезпечують масштабованість, безпеку та інтеграцію з зовнішніми сервісами, що створює основу для ефективної реалізації автоматизованого робочого місця адміністратора автосалону.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОЄКТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ АРМ АДМІНІСТРАТОРА ДЛЯ ПРОДАЖУ АВТО

На етапі реалізації розробленої інформаційної системи АРМ адміністратора було здійснено створення повноцінного вебзастосунку на базі архітектури клієнт-сервер, що забезпечує інтегровану взаємодію між користувачами системи, зокрема адміністраторами, бухгалтерами та клієнтами.

Розробка здійснювалась із використанням мови програмування Java 17 у поєднанні з фреймворком Spring Boot, що забезпечив зручне створення REST API для взаємодії між клієнтською та серверною частинами. Для реалізації бази даних обрана СКБД PostgreSQL, що гарантує цілісність і масштабованість збереження даних. Застосунок підтримує зв’язок із зовнішніми сервісами, зокрема банківськими API, поштовими сервісами та типовим рішенням бухгалтерського обліку [7], [9], [6], [12].

Реалізовано такі основні модулі:

- модуль аутентифікації та авторизації;
- каталог автомобілів для менеджерів і клієнтів;
- інтерфейс перегляду, фільтрації, редагування замовлень;
- адміністративна панель для керування персоналом та правами доступу;
- модуль статистики продажів і замовлень;
- інтеграція з бухгалтерською системою для синхронізації платіжної інформації.

Ключові фрагменти програмного коду для авторизації та сутності авто було наведено у додатку Б.

Відповідно до потреби навчання адміністраторів, котрі будуть мати потребу у розгортанні системи, була створена детальна інструкції, та наведена у Додатку А.

Нижче на Рисунках 3.1-3.5 буде наведено вигляд вебзастосунку "Адміністратор" для продажу авто:

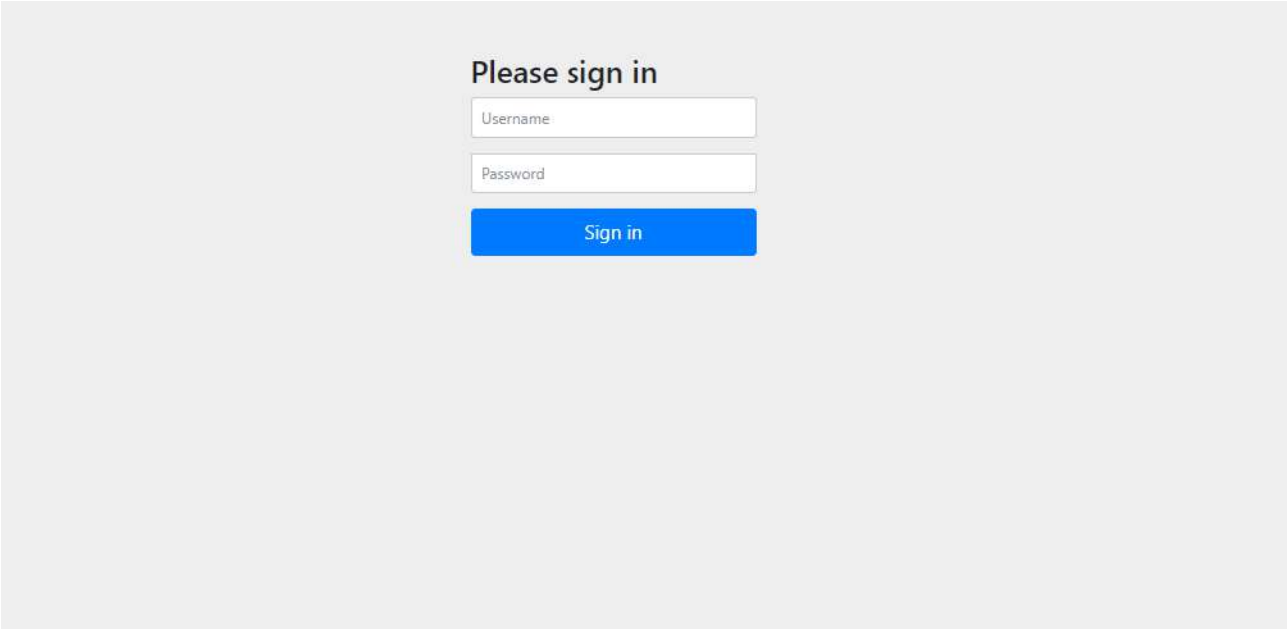

Рисунок 3.1 — Сторінка авторизації

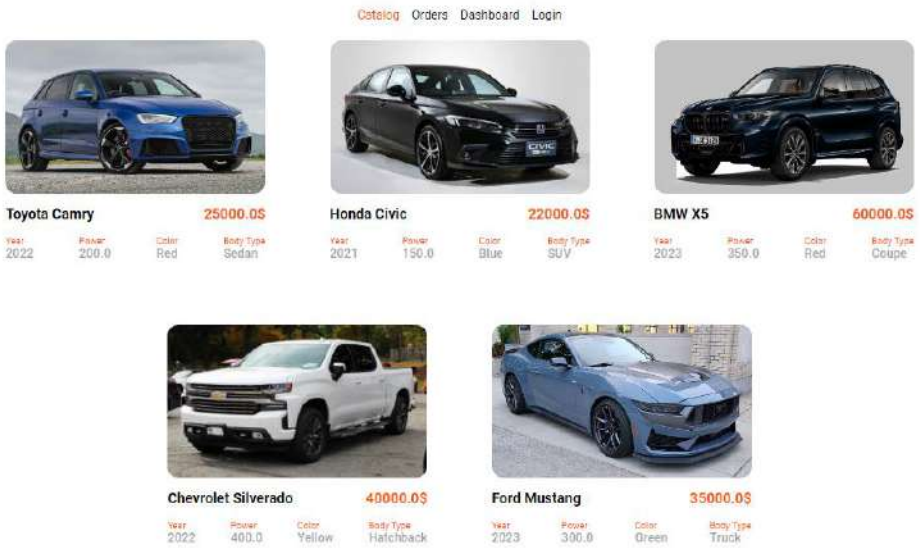

Рисунок 3.2 — Каталог автомобілів

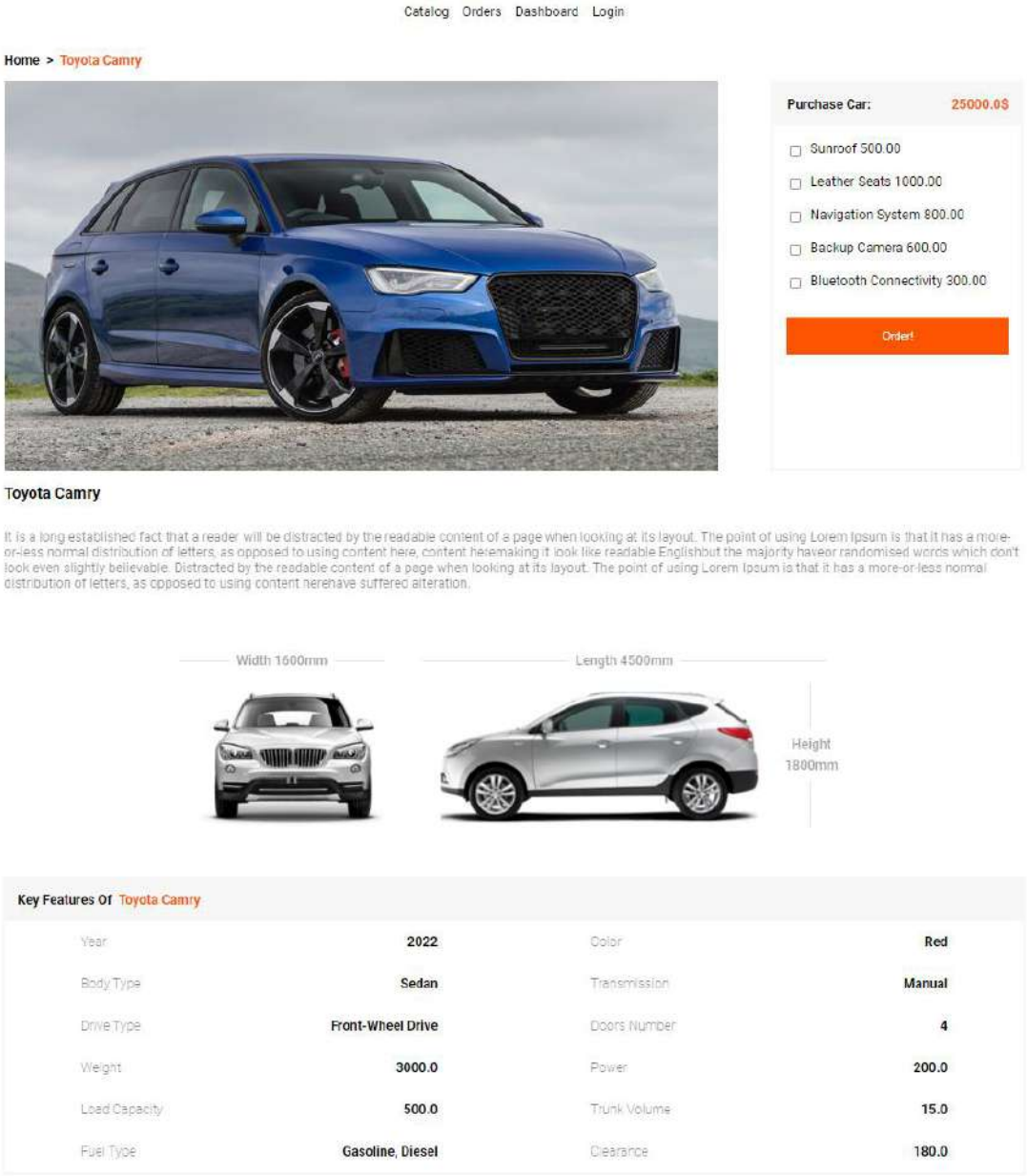

Рисунок 3.3 — Інформація про авто та додаткові опції

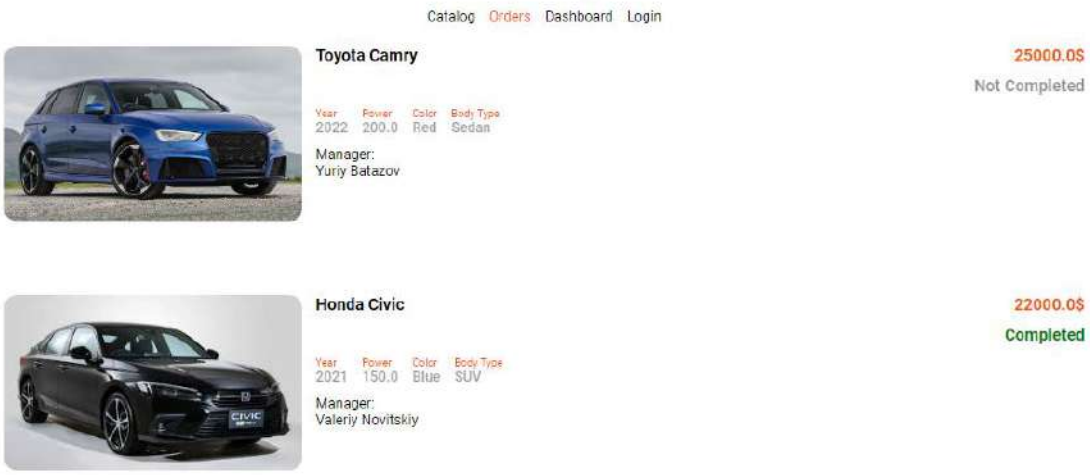

Рисунок 3.4 — Історія замовлень

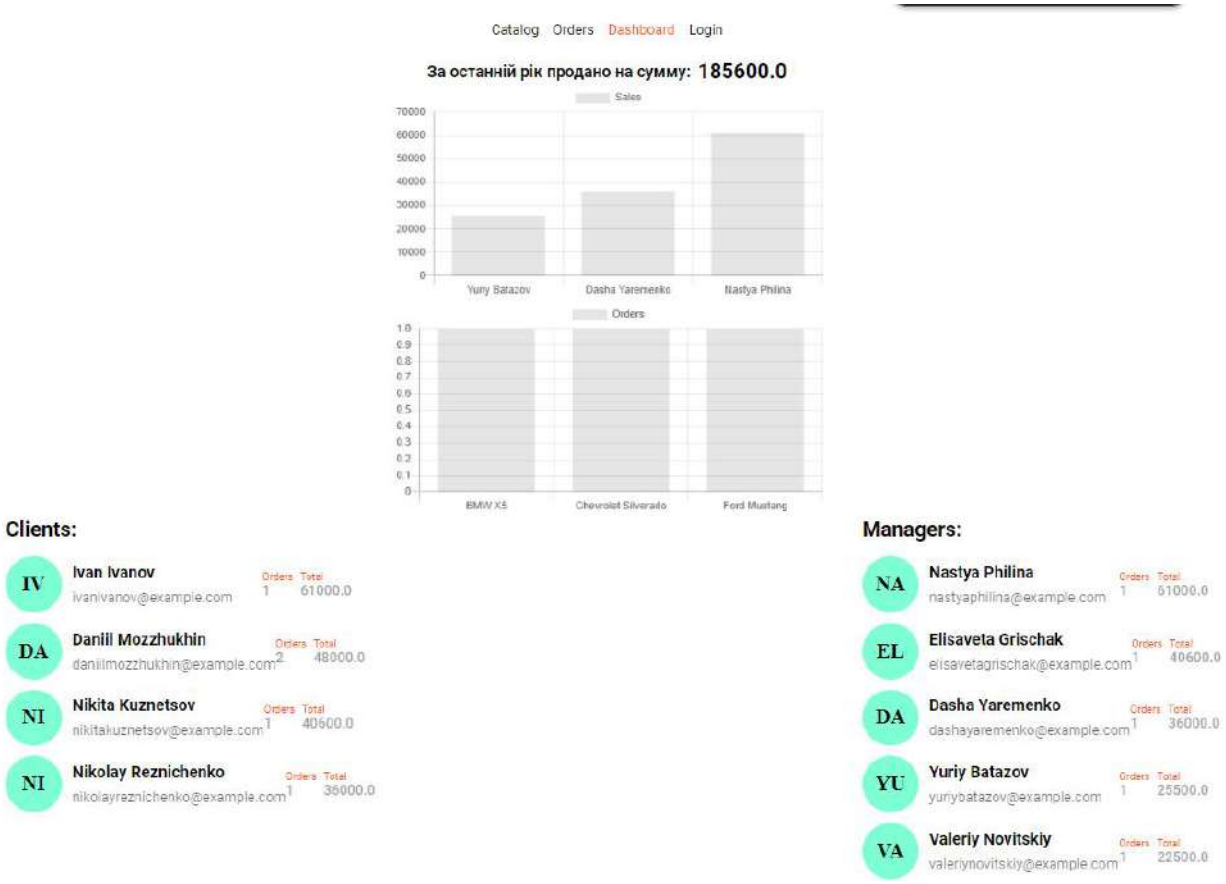

Рисунок 3.5 — Статистика продажів

Система автоматично передає інформацію про оплати, підтвердження замовлень і фінансову звітність до типової бухгалтерської системи через попередньо налаштовані канали обміну або API. Це забезпечує цілісність фінансового обліку без дублювання операцій, дозволяє уникати ручного введення даних і зменшує ризики помилок.

Таким чином, реалізована інформаційна система повністю відповідає сформульованим у попередніх розділах вимогам, охоплює основні бізнес-процеси автосалону та сприяє підвищенню ефективності управління й обліку.

Висновок до розділу 3:

Було реалізовано інформаційну систему автоматизованого робочого місця адміністратора для автосалону у вигляді вебзастосунку з клієнт-серверною архітектурою. Система включає всі необхідні модулі для ефективного

управління даними, замовленнями, користувачами, а також забезпечує інтеграцію з бухгалтерським обліком і зовнішніми сервісами. Функціонал реалізовано із застосуванням сучасних технологій, що дозволяє підвищити продуктивність, прозорість і масштабованість бізнес-процесів автосалону. Було проведене загальне тестування з ціллю виявлення можливих багів та проблем, після виявлення вони були виправлені у фінальній версії застосунку. Реалізація відповідає вимогам, сформульованим на етапах проєктування, та підтверджує доцільність обраних рішень.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ

Розробка та впровадження інформаційної системи АРМ адміністратора для продажу автомобілів передбачає участь персоналу, що працює в офісному середовищі за комп'ютерами, зокрема адміністраторів, бухгалтерів, менеджерів з продажу. Тому при організації робочих місць і проведенні робіт важливо дотримуватися норм і вимог законодавства з охорони праці. Основні вимоги до безпечної організації праці регламентуються такими документами: Закон України «Про охорону праці»; Державні санітарні правила і норми ДСанПіН 3.3.2.007-98 «Гігієнічні вимоги до відеодисплейних терміналів»; ДСТУ EN ISO 9241-5:2002 «Ергономічні вимоги до офісної роботи з відеотерміналами»; Правила пожежної безпеки в Україні (НАПБ А.01.001-2004). Ці документи визначають вимоги до облаштування офісного простору, використання електронної техніки, організації праці з ПЕОМ, санітарно-гігієнічні норми, тривалість роботи, організацію відпочинку та інше [13].

Робоче місце адміністратора передбачає тривале перебування за комп'ютером, тому особливу увагу слід приділяти ергономіці. Висота столу повинна бути 720–750 мм, крісло регулюється по висоті й нахилу спинки. Відстань від очей до монітора має становити 60–70 см, кут огляду — 15–20° нижче рівня очей. Освітленість робочої зони повинна бути не менше 300 люкс при денному світлі й не менше 500 люкс при штучному. Не допускається відблисків на моніторі, джерела світла мають бути розташовані збоку. На моніторі має бути встановлений режим фільтрації синього світла та автоматичне регулювання яскравості. Для зменшення навантаження на зір і опорно-руховий апарат працівникам рекомендується кожні 60 хвилин роботи за ПК робити 10–15-хвилинну перерву, виконувати вправи для очей та гімнастику для спини, змінювати положення тіла [16].

Серверне та комп’ютерне обладнання потребує дотримання норм пожежної безпеки. Не допускається перевантаження електричних мереж. Всі кабелі мають бути сертифіковані. Обов’язкова наявність вогнегасника в офісному приміщенні. Електроприлади повинні мати справні штепсельні розетки, заземлення та не повинні працювати без нагляду. У приміщенні мають бути розроблені та розміщені на видному місці плани евакуації, інструкції з пожежної безпеки, номери аварійних служб [14].

При роботі з комп’ютерною технікою працівник перебуває в зоні дії електроструму низької напруги. Застосування справних і сертифікованих джерел живлення, забезпечення захисного заземлення та автоматичного вимкнення при короткому замиканні, періодична перевірка кабелів, штепселів і розеток, проведення інструктажів з електробезпеки є необхідними заходами захисту.

Тривала робота з інформацією та спілкування з клієнтами може спричиняти психоемоційне напруження. Тому важливо уникати понаднормової роботи, забезпечити комфортну атмосферу в колективі, проводити тренінги зі стресостійкості. Тривалість робочого дня не повинна перевищувати 8 годин. Згідно з вимогами охорони праці, робоча зміна повинна включати обідню перерву тривалістю не менше 30 хвилин, дозволяється додатковий 10-хвилинний відпочинок щогодини при роботі за монітором, забороняється залучення до понаднормової роботи без письмової згоди працівника [15].

Врахування вимог охорони праці є необхідною умовою для безпечного та ефективного функціонування інформаційної системи АРМ адміністратора. Дотримання встановлених норм дозволяє зменшити вплив шкідливих чинників, підвищити продуктивність праці та забезпечити належний рівень захисту життя і здоров’я працівників.

Висновок до розділу 4:

У цьому розділі було розглянуто основні вимоги з охорони праці, що стосуються організації безпечного та комфортного робочого місця

адміністратора автосалону. Визначено нормативно-правову базу, яка регламентує умови праці в офісному середовищі, а також проаналізовано ергономічні, санітарно-гігієнічні, електробезпечні та пожежобезпечні аспекти роботи з комп’ютерною технікою. Дотримання зазначених норм сприяє зниженню ризиків для здоров’я працівників, підвищенню ефективності роботи та забезпеченню безперервного функціонування автоматизованої інформаційної системи. Раціональна організація праці та дотримання техніки безпеки є невіддільною складовою сучасного цифрового середовища.

ВИСНОВОК

У результаті виконання дипломної роботи було реалізовано повний цикл розробки автоматизованої інформаційної системи для потреб автосалону, зокрема — автоматизованого робочого місця адміністратора. Проведений аналіз предметної області дозволив виявити ключові проблеми в організації бізнес-процесів: фрагментація обліку, ручна обробка даних, відсутність інтеграції між підрозділами, що знижувало рентабельність і створювало додаткові ризики для бізнесу.

У ході роботи:

- досліджено організаційну структуру автосалону та проаналізовано обов’язки основних підрозділів;

- виявлено основні проблеми чинної організації роботи адміністратора;

- сформульовано функціональні та нефункціональні вимоги до інформаційної системи;

- обґрунтовано вибір технологічного стеку: Java, Spring Boot, PostgreSQL, та допоміжних засобів розробки й адміністрування;

- створено концептуальну, логічну та фізичну моделі бази даних;

- реалізовано клієнт-серверну архітектуру з вебінтерфейсом для адміністратора;

- розроблено повноцінний вебзастосунок з модулями аутентифікації, замовлень, статистики, управління користувачами;

- забезпечено інтеграцію з типовим бухгалтерським рішенням для автоматизованого фінансового обліку;

- проведено тестування реалізованого функціоналу, що підтвердило коректність роботи основних компонентів системи;

- розглянуто нормативні вимоги з охорони праці, пов’язані з організацією безпечного робочого місця адміністратора.

У результаті розроблена система дозволяє централізовано управляти клієнтськими запитами, контролювати замовлення та платежі, обробляти звітність і забезпечує ефективну взаємодію адміністратора та бухгалтера. Це сприятиме підвищенню продуктивності персоналу, зниженню операційних витрат і покращенню якості обслуговування клієнтів автосалону.

Таким чином, усі поставлені у вступі завдання були успішно виконані, а створена інформаційна система повністю відповідає вимогам сучасного бізнесу в сфері автомобільних продажів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Марченко А.В. Проектування інформаційних систем СумДУ, 2016. – 90 с. [Електронний ресурс]. – https://duikt.edu.ua/uploads/l_144_42481385.pdf

2. Ременяк Л.В. Проектування інформаційних систем: конспект лекцій. – Одеса: ОДЕУ, 2016. – 152 с. http://eprints.library.odeku.edu.ua/734/

3. Pressman, R. S. Software Engineering: A Practitioner's Approach. – 8th ed. – McGraw-Hill, 2014. – 960 p. https://invent.ilmkidunya.com/images/Section/12.pdf

4. Книрік Н.Р. Правила трансформації концептуальної схеми баз даних. Лекція 5.

5. Фаулер М. UML. Основи: навчальний посібник. – 3-тє видання. – Київ: Символ-Плюс, 2005. – 208 с. https://k0d.cc/storage/books/UML/uml_osnovy_3-e_izd.pdf

6. Книрік Н. Р. DML – Data Manipulation Language. Лекція 3

7. Vaughn, D. Implementing Domain-Driven Design. – Addison-Wesley, 2013. – 560 p. https://dl.ebooksworld.ir/motoman/AW.Implementing.Domain-Driven.Design.www.EBooksWorld.ir.pdf

8. Kleppmann, M. Designing Data-Intensive Applications. – O'Reilly Media, 2017. – 616 p. https://www.oreilly.com/library/view/designing-data-intensive-applications/9781491903063/

9. Васильєв О. Програмування мовою Java. – Київ: ФОП «Брайт Стар Паблішинг», 2020. – 420 с.

10. Мартін Р. Чистий код: створення, аналіз і рефакторинг. – Львів: Видавництво Старого Лева, 2019. – 464 с.

11. Walls, C. Spring in Action. – 6th ed. – Manning Publications, 2022. – 520 p.

12. Bloch, J. Effective Java. – 3rd ed. – Addison-Wesley, 2018. – 416 p. https://kea.nu/files/textbooks/new/Effective%20Java%20%282017%2C%20Addison-Wesley%29.pdf

13. Закон України «Про охорону праці» № 2694-XII від 14.10.1992 р. https://zakon.rada.gov.ua/laws/show/2694-12#Text

14. НАПБ А.01.001-2004: Правила пожежної безпеки в Україні. – МВС України. https://online.budstandart.com/ua/catalog/doc-page?id_doc=29321

15. Шеремет А.М. Охорона праці в галузі: навч. посібник. – Київ: Каравела, 2016. – 256 с. http://moodle.nati.org.ua/pluginfile.php/14074/mod_resource/content/1/Ohorona_pratsi.pdf

16. ДСанПіН 3.3.2.007-98: Гігієнічні вимоги до відеодисплейних терміналів персональних електронно-обчислювальних машин. – МОЗ України, 1998. https://zakon.rada.gov.ua/rada/show/v0007282-98

17. Коваленко, С. П., Петренко, І. М. Автоматизація управління підприємством: навч. посібник. – Львів: ЛНУ імені Івана Франка, 2020. – 320 с. https://stlnau.in.ua/samoosvita/item/2024/iit241010.pdf

18. Бази даних та їх проектування. – Київ: Видавництво "Наукова думка", 2017. – 280 с.

19. Мельник, О. О. Розробка інформаційних систем на базі сучасних технологій. – Харків: ХНУРЕ, 2019. – 200 с.

ДОДАТОК А

Інструкція адміністратора

Цей додаток допоможе вам створити пусту базу даних для автосалону і відновити її структуру та дані з резервної копії (файлу SQL), який знаходиться в кореневій папці проєкту.

Перший крок. Встановлення PostgreSQL і pgAdmin

1. Завантажте та встановіть PostgreSQL з офіційного сайту: https://www.postgresql.org/download/

2. Запам’ятайте пароль користувача postgres, який задаєте під час інсталяції.

3. Встановіть pgAdmin — графічний інтерфейс для керування PostgreSQL, який можна завантажити тут: https://www.pgadmin.org/download/

Другий крок. Підключення до PostgreSQL у pgAdmin

1. Запустіть pgAdmin.

2. Підключіться до вашого PostgreSQL сервера (зазвичай PostgreSQL 12/13/14).

3. Введіть пароль користувача postgres при запиті.

Третій крок. Створення пустої бази даних

1. В лівій панелі pgAdmin клацніть правою кнопкою на Databases → Create → Database

2. Введіть назву бази даних CarDealership.

3. Власником бази залиште postgres (або вкажіть відповідного користувача).

4. Натисніть Save для створення пустої бази.

Четвертий крок. Відновлення бази з файлу резервної копії

1. Клацніть правою кнопкою на щойно створеній базі CarDealership.

2. Оберіть пункт Restore.

3. В полі вибору файлу вкажіть шлях до файлу резервної копії (SQL-скрипта) з кореневої папки вашого проєкту CarDealership.sql.

4. Переконайтеся, що формат файлу обраний відповідно до типу файлу (Plain або Custom).

5. Натисніть кнопку Restore.

6. Процес займе декілька секунд, після чого база матиме всю структуру таблиць та початкові дані.

П’ятий крок. Перевірка результату

1. Розгорніть базу в pgAdmin → Schemas → public → Tables.

2. Переконайтеся, що всі необхідні таблиці створені.

3. За потреби виконайте SQL-запити для перевірки даних, наприклад: SELECT * FROM cars;

Шостий крок. Резервне копіювання бази даних (бекап)

1. Клацніть правою кнопкою на базі car_dealership.

2. Оберіть Backup.

3. Вкажіть місце і назву файлу, куди буде збережено резервну копію.

4. Виберіть формат (рекомендується Custom).

5. Натисніть Backup.

Примітки

Для роботи з файлами резервних копій краще використовувати формат Custom, але pgAdmin підтримує і plain SQL-файли.

Якщо виникають помилки при відновленні, переконайтеся, що створена база пуста і що користувач має достатні права.

Якщо потрібно, можна автоматизувати процес відновлення через командний рядок pg_restore або psql.

ДОДАТОК Б

Фрагменти лістингу

Посилання на повний проект: https://github.com/SofaNinja/DBCarDealership

SecurityConfiguration.java:

```
package com.example.dbcardealership.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.authentication.AuthenticationProvider
;
import
org.springframework.security.authentication.dao.DaoAuthenticationP
rovider;
import
org.springframework.security.config.annotation.method.configuratio
n.EnableMethodSecurity;
import
org.springframework.security.config.annotation.web.builders.HttpSe
curity;
import
org.springframework.security.config.annotation.web.configuration.E
nableWebSecurity;
import
org.springframework.security.config.annotation.web.configurers.Abs
tractHttpConfigurer;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import
org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import com.example.dbcardealership.service.UserDetailsService;

@EnableWebSecurity
@EnableMethodSecurity
@Configuration
public class SecurityConfiguration {
    @Bean
    public UserDetailsService userDetailsService(){
        return new
com.example.dbcardealership.service.UserDetailsService();
    }

    @Bean
```

```
    public     AuthenticationProvider     authenticationProvider(){
        DaoAuthenticationProvider       provider       =       new
DaoAuthenticationProvider();
        provider.setUserDetailsService(userDetailsService());
        provider.setPasswordEncoder(passwordEncoder());
        return                                                 provider;
    }

    @Bean
    public   SecurityFilterChain   securityFilterChain(HttpSecurity
http)                throws                Exception                {
        return                                                      http
                .csrf(AbstractHttpConfigurer::disable)
                .authorizeHttpRequests(auth         ->         auth

.requestMatchers("/dashboard/**").authenticated()

.requestMatchers("/order/**").authenticated()
                        .anyRequest().permitAll()
                )
                .formLogin(formLogin                                ->
formLogin.defaultSuccessUrl("/car"))
                .build();
    }
    @Bean
    public            PasswordEncoder            passwordEncoder(){
        return            new            BCryptPasswordEncoder(5);
    }
}
```

CarController.java:

```
    package                com.example.dbcardealership.controller;

import                              lombok.RequiredArgsConstructor;
import                  org.springframework.stereotype.Controller;
import                              org.springframework.ui.Model;
import                  org.springframework.web.bind.annotation.*;
import                 com.example.dbcardealership.model.NewCarDTO;
import
com.example.dbcardealership.repository.AdditionalOptionRepository;
import              com.example.dbcardealership.service.CarService;

@Controller
@RequiredArgsConstructor
@RequestMapping("/car")
public              class              CarController              {
    private          final          CarService          carService;
    private            final            AdditionalOptionRepository
additionalOptionRepository;
```

```
    @GetMapping
    public String getCarList(Model model) {
        model.addAttribute("carList", carService.getAllCars());
        return "home";
    }

    @GetMapping("/{car_id}")
    public String getCarInformationById(@PathVariable("car_id") int
car_id, Model model) {
        System.out.println(car_id);
        var data = carService.getCarById(car_id);
        model.addAttribute("carData", data);
        model.addAttribute("options",
additionalOptionRepository.getOptionList());

        return "car";
    }

    @PostMapping("/new")
    public String newCar(@RequestParam String model,
                         @RequestParam Integer year,
                         @RequestParam Double price,
                         @RequestParam String color,
                         @RequestParam Integer numberOfDoors,
                         @RequestParam Integer capacity,

                         @RequestParam Integer carWidth,
                         @RequestParam Integer carLength,
                         @RequestParam Integer carHeight,

                         @RequestParam Integer wheelbase,
                         @RequestParam Double power,

                         @RequestParam Integer bodyTypeId,
                         @RequestParam Integer transmissionTypeId,
                         @RequestParam Integer[] fuelTypeId,
                         @RequestParam Integer driveTypeId
    ) {
        carService.addNewCar(new NewCarDTO(model, year, price,
color, numberOfDoors, capacity, carWidth, carLength, carHeight,
wheelbase, power, bodyTypeId, transmissionTypeId, fuelTypeId,
driveTypeId));
        return "redirect:/car";
    }
}
```

CarRepository.java:

```
package com.example.dbcardealership.repository;

import org.springframework.data.jpa.repository.JpaRepository;
```

```
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import com.example.dbcardealership.entity.Car;
import com.example.dbcardealership.model.CarDTO;
import com.example.dbcardealership.model.CarSmallDTO;

import java.util.List;

public interface CarRepository extends JpaRepository<Car, Integer>
{

    @Query(value = """
select
C.model,
C.year,
C.price,
BT.name as bodyType,
FT.name as fuelType,
TT.name as transmissionType,
D.name as driveType,
C.number_of_doors as doorsNumber,
C.capacity,
C.dimensions,
C.wheelbase,
C.engine_volume as engineVolume,
C.power,
C.weight,
C.clearance,
C.load_capacity as loadCapacity,
C.trunk_volume as trunkVolume,
C.color,
C.image
from cars C
left join drive_types D ON D.drive_type_id = C.drive_type_id
left join car_fuel_types CFT on CFT.car_id = C.car_id
left join fuel_types FT on FT.fuel_type_id = CFT.fuel_type_id
left join transmission_types TT on TT.transmission_type_id =
C.transmission_type_id
left join body_types BT on BT.body_type_id = C.body_type_id
where C.car_id = ?1
""", nativeQuery = true)
    List<CarDTO> getCarInformationByCarId(int carId);

    @Query(value = """
select
    C.car_id as id,
    C.model,
    C.price,
    C.image,
    C.year,
    C.power,
    C.color,
```

```
    BT.name as bodyType
from cars C
left join body_types BT on BT.body_type_id = C.body_type_id;
""", nativeQuery = true)
    List<CarSmallDTO> getAllCars();

    @Modifying
    @Query(value = """
insert into cars (
                  model,
                  year,
                  price,
                  body_type_id,
                  transmission_type_id,
                  drive_type_id,
                  number_of_doors,
                  capacity,
                  dimensions,
                  wheelbase,
                  max_torque,
                  engine_volume,
                  power,
                  weight,
                  clearance,
                  load_capacity,
                  trunk_volume,
                  color)
VALUES (?1, ?2, ?3, ?4, ?5, ?6, ?7, ?8, ?9, ?10, 200, 4.5, ?11, 3000,
170, 450, 15, ?12)
""", nativeQuery = true)
    Integer insertCar(
            String model,
            Integer year,
            Double price,
            Integer bodyTypeId,
            Integer transmissionTypeId,
            Integer driveTypeId,
            Integer numberOfDoors,
            Integer capacity,
            String dimensions,
            Integer wheelbase,
            Double power,
            String color
    );

    @Modifying
    @Query(value = """
insert into car_fuel_types (car_id, fuel_type_id) VALUES (?1, ?2)
""", nativeQuery = true)
    void insertFuelTypes(Integer carId, Integer fuelTypeId);
}
```

CarService.java:

```
package com.example.dbcardealership.service;

import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.example.dbcardealership.model.*;
import
com.example.dbcardealership.repository.AdditionalOptionRepository;
import com.example.dbcardealership.repository.CarRepository;

import java.util.Arrays;
import java.util.List;

@Service
@RequiredArgsConstructor
public class CarService {
    private final CarRepository carRepository;
    private final AdditionalOptionRepository
additionalOptionRepository;

    public List<CarSmallDTO> getAllCars() {
        return carRepository.getAllCars();
    }

    List<Element> getFuelTypeList() {
        return additionalOptionRepository.getFuelTypeList();
    }
    List<Element> getDriveTypeList() {
        return additionalOptionRepository.getDriveTypeList();
    }

    List<Element> getTransmissionTypeList() {
        return
additionalOptionRepository.getTransmissionTypeList();
    }

    @Transactional
    public void addNewCar(NewCarDTO newCar) {
        var sizes = newCar.getCarLength() + "mm x " +
newCar.getCarHeight() + "mm x " + newCar.getCarWidth() + "mm";
        var carId = carRepository.insertCar(newCar.getModel(),
newCar.getYear(), newCar.getPrice(), newCar.getBodyTypeId(),
newCar.getTransmissionTypeId(), newCar.getDriveTypeId(),
newCar.getNumberOfDoors(), newCar.getCapacity(), sizes,
newCar.getWheelbase(), newCar.getPower(), newCar.getColor());
        for (Integer element : newCar.getFuelTypeId()) {
            System.out.println(element);
            carRepository.insertFuelTypes(carId, element);
        }
```

```
    }

    public CarResponse getCarById(int id) {
        var result = carRepository.getCarInformationByCarId(id);
        var fuelTypes =
result.stream().map(CarDTO::getFuelType).toArray();
        var fuelTypeString =
Arrays.toString(fuelTypes).replace("[", "").replace("]", "");
        var sizes =
Arrays.stream(result.get(0).getDimensions().split("x")).map(String
::strip).toList();

        return new CarResponse()
                .setImage(result.get(0).getImage())
                .setModel(result.get(0).getModel())
                .setYear(result.get(0).getYear())
                .setPrice(result.get(0).getPrice())
                .setBodyType(result.get(0).getBodyType())
                .setFuelType(fuelTypeString)

.setTransmission(result.get(0).getTransmissionType())
                .setDriveType(result.get(0).getDriveType())
                .setDoorsNumber(result.get(0).getDoorsNumber())
                .setCapacity(result.get(0).getCapacity())
                .setCarHeight(sizes.get(1))
                .setCarWidth(sizes.get(2))
                .setCarLength(sizes.get(0))
                .setWheelbase(result.get(0).getWheelbase())
                .setEngineVolume(result.get(0).getEngineVolume())
                .setPower(result.get(0).getPower())
                .setWeight(result.get(0).getWeight())
                .setClearance(result.get(0).getClearance())
                .setLoadCapacity(result.get(0).getLoadCapacity())
                .setTrunkVolume(result.get(0).getTrunkVolume())
                .setColor(result.get(0).getColor());
    }
}
```