ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему «Мобільний додаток для підвищення ефективності бізнес-процесів в автомобільному салоні мовою програмування Java»

на здобуття освітнього ступеня бакалавра

зі спеціальності 126 Інформаційні системи та технології

освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень.
Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ _____
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувачка вищої освіти гр. ІСД-43
Надія ЯРМАК
Ім'я, ПРІЗВИЩЕ

Керівник:
науковий ступінь, вчене звання
Владислав ЗАВАЦЬКИЙ
Ім'я, ПРІЗВИЩЕ

Рецензент:
науковий ступінь, вчене звання
Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедрою ІСТ

_____Каміла СТОРЧАК

«____» _____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ярмак Надія Віталіївна
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Мобільний додаток для підвищення ефективності бізнес-процесів в автомобільному салоні мовою програмування Java

керівник кваліфікаційної роботи Владислав ЗАВАЦЬКИЙ,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» 02 2025р. № 56

2. Строк подання кваліфікаційної роботи «30» 05 2025 р.

3. Вихідні дані до кваліфікаційної роботи: офіційна документація Java, науково-технічна література, дані про існуючі системи підвищення ефективності бізнес-процесів у відкритих наукових джерелах.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз поточного стану та ефективності бізнес-процесів у автомобільному салоні
2. Огляд існуючих інструментів та технологій для підвищення ефективності бізнес-процесів
3. Розробка додатку мовою Java для підвищення ефективності бізнес-процесів

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «24» 02 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	01.03.2025 р.	
2	Дослідження поточного стану та ефективності бізнес-процесів на підприємстві автомобільного салону	15.03.2025 р.	
3	Аналіз існуючих рішень для підвищення ефективності бізнес-процесів	17.03.2025 р.	
4	Постановка технічного завдання, обґрунтований вибір технологічного стеку та проєктування архітектури мобільного додатку	4.04.2025 р.	
5	Реалізація та тестування мобільного додатку мовою програмування Java	16.05.2025 р.	
6	Розробка демонстраційних матеріалів	26.05.2025 р.	
7	Попередній захист дипломної роботи	27.05.2025 р.	
8	Подання роботи в деканат	30.05.2025 р.	

Здобувач(ка) вищої освіти _____ Надія ЯРМАК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Владислав ЗАВАЦЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 72 стор., 59 рис., 20 джерел.

Мета роботи – розробити та впровадити мобільний застосунок для цифровізації бізнес-процесів автосалону *Ontario Quality Motors* з метою підвищення ефективності управління, покращення клієнтського сервісу та автоматизації внутрішніх операцій.

Об'єкт дослідження – бізнес-процеси автосалону, пов'язані з продажем, обслуговуванням автомобілів та взаємодією з клієнтами.

Предмет дослідження – технології розробки мобільних інформаційних систем, архітектурні рішення, інструменти програмної інженерії та засоби цифровізації бізнесу в автомобільній сфері.

Короткий зміст роботи: У роботі проаналізовано поточний стан цифровізації підприємства *Ontario Quality Motors*, розглянуто світовий досвід цифрових трансформацій у сфері автоторгівлі, описано вибір технологій та архітектури системи. Детально представлено реалізацію мобільного застосунку на основі Java, Spring Boot, Android Studio та PostgreSQL, а також систему безпеки, аналітику продажів і функціонал для трьох ролей користувачів. Завершальний розділ присвячено тестуванню програмного забезпечення та впровадженню у робоче середовище.

КЛЮЧОВІ СЛОВА: МОБІЛЬНИЙ ЗАСТОСУНОК, ЦИФРОВІЗАЦІЯ, БІЗНЕС-ПРОЦЕСИ, АВТОСАЛОН, SPRING BOOT, ANDROID STUDIO, POSTGRESQL, ІНФОРМАЦІЙНА СИСТЕМА, АВТОМАТИЗАЦІЯ, ЦИФРОВА ТРАНСФОРМАЦІЯ.

ABSTRACT

Textual part of the qualification work for obtaining a bachelor's degree: 72 pages, 59 figs., 20 sources.

The purpose of the work is to develop a mobile application for optimization and digital transformation of business processes of the car dealership, which will provide effective management of customer requests, vehicle accounting, staff work and sales analytics.

The object of research is business processes of customer service and fleet management in the activities of a car dealership.

The subject of the study is methods and tools for developing a mobile application to automate the processes of interaction between customers, administrators and sellers of a car dealership.

Summary of the work: The paper considers modern approaches to the digital transformation of business in the field of car sales. The requirements for the mobile application are analyzed, the architecture is defined, the database, server logic and client interface are developed. Three user roles are implemented: administrator, seller and client. Customers can view cars, leave requests and form a list of favorites. The administrator can manage the fleet, create sellers, assign them to applications and view sales analytics. Sellers process the applications assigned to them and update their statuses. The application is implemented using Java, Android SDK, Spring Boot, PostgreSQL, Retrofit, and MVVM architecture.

KEYWORDS: MOBILE APPLICATION, CAR DEALERSHIP, DIGITAL TRANSFORMATION, APPLICATIONS, DATABASE, BUSINESS AUTOMATION.

ЗМІСТ

ВСТУП

Актуальність теми дослідження зумовлена необхідністю цифрової трансформації бізнес-процесів в умовах динамічного розвитку інформаційних технологій. В умовах жорсткої конкуренції на ринку автомобільної торгівлі особливого значення набуває впровадження сучасних цифрових рішень, які сприяють підвищенню ефективності управління, оптимізації внутрішніх процесів, автоматизації обробки клієнтських заявок та поліпшенню якості сервісного обслуговування. Мобільні застосунки як інструмент цифрової взаємодії між підприємством і клієнтами забезпечують гнучкість, доступність, зручність користування та високу швидкість обміну інформацією. Саме тому розробка мобільного застосунку для підприємства, яке спеціалізується на продажу та обслуговуванні вживаних автомобілів, є актуальним напрямом дослідження, що має значну практичну цінність.

Метою дипломної роботи є розробка мобільного застосунку для підприємства Ontario Quality Motors з метою цифровізації ключових бізнес-процесів, зокрема управління клієнтською базою, заявками, автопарком і взаємодією персоналу, а також підвищення продуктивності та якості обслуговування клієнтів. Для досягнення поставленої мети в роботі реалізовано такі завдання: проведено аналіз рівня цифровізації підприємства та виявлено існуючі проблеми в управлінні; вивчено міжнародний досвід цифровізації автосалонів; визначено технологічний стек і мову програмування для реалізації програмного продукту; сформульовано вимоги до бази даних і системи безпеки; спроєктовано архітектуру мобільного застосунку відповідно до принципів MVVM; реалізовано основні функціональні модулі застосунку; здійснено його тестування за допомогою Mockito та Postman; сформульовано рекомендації щодо подальшого впровадження.

Об’єктом дослідження виступають бізнес-процеси підприємства Ontario Quality Motors, пов’язані з продажем транспортних засобів та обслуговуванням

клієнтів, а предметом — методи, засоби та інструменти розробки мобільного застосунку для оптимізації зазначених процесів. Наукова новизна роботи полягає у створенні комплексного рішення на основі сучасних технологій, таких як Java, Android Studio, Spring Boot, PostgreSQL, Firebase, Retrofit та архітектури MVVM, що забезпечує інтеграцію з зовнішніми сервісами та високий рівень безпеки.

Апробація результатів дослідження здійснювалась шляхом участі у II Всеукраїнській науково-технічній конференції «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу» з поданням тез доповіді на тему «Особливості розробки мобільного застосунку для управління та оптимізації бізнес-процесів автомобільного салону: сучасні підходи та інструменти цифрової трансформації». Практична значущість отриманих результатів полягає в можливості впровадження розробленого мобільного застосунку на реальному підприємстві з метою підвищення ефективності його роботи, конкурентоспроможності та клієнтоорієнтованості.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПОТОЧНОГО СТАНУ ЦИФРОВІЗАЦІЇ НА ПІДПРИЄМСТВІ “ONTARIO QUALITY MOTORS”

1.1 Аналіз поточного використання цифрових технологій на підприємстві

Підприємство "Ontario Quality Motors" є офіційним автомобільним дилерським центром, що розташоване в Торонто і спеціалізується на реалізації вживаних автомобілів різних марок, моделей та цінових категорій.

Основним видом діяльності компанії є продаж транспортних засобів, які проходять ретельну перевірку перед реалізацією, що гарантує їхню високу якість та надійність. Крім того, підприємство пропонує комплексне технічне обслуговування, включаючи діагностику, ремонт та регулярне сервісне обслуговування автомобілів, що спрямоване на забезпечення максимальної задоволеності клієнтів.

Рис. 1.1 Фасад підприємства Ontario Quality Motors[7]

Особливістю діяльності компанії є орієнтація на гнучкі умови оплати, які включають як готівковий розрахунок, так і кредитування чи розстрочку, що

дозволяє зробити придбання автомобілів доступним для широкого кола споживачів. Одним із ключових завдань "Ontario Quality Motors" є впровадження високих стандартів обслуговування клієнтів, що включає професійний підхід, надійність сервісу та персоналізований підхід до потреб кожного замовника.

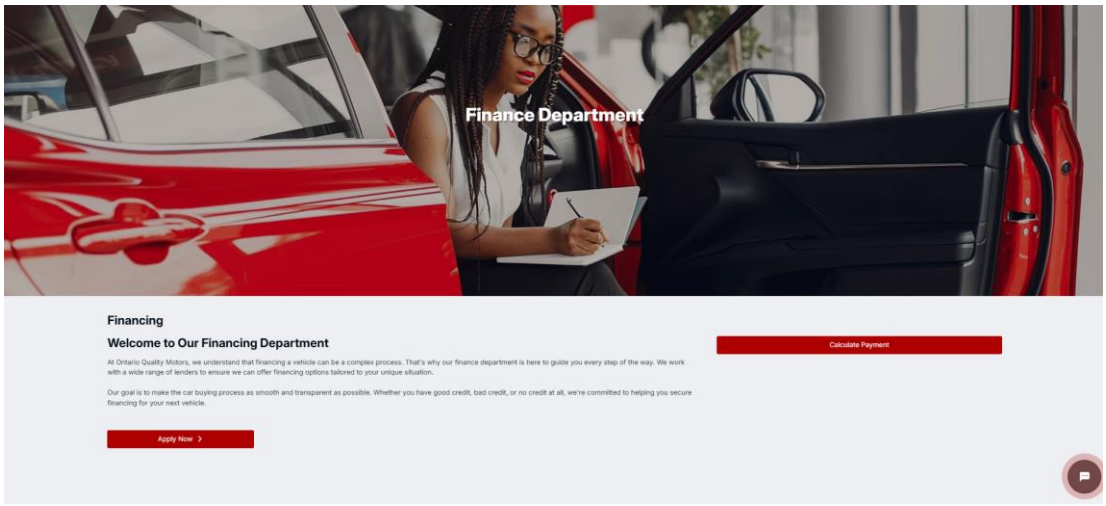


Рис. 1.2 Інтерфейс фінансового відділу Ontario Quality Motors[7]

Для підтримки конкурентоспроможності та високого рівня обслуговування клієнтів компанія активно впроваджує сучасні цифрові технології, зокрема спеціалізовані програмні продукти Hillz Dealer і DealerTrack, а також корпоративний веб-сайт для клієнтів.

Hillz Dealer є інтегрованою системою управління, розробленою для автомобільних дилерів, що забезпечує централізоване зберігання даних та управління бізнес-процесами підприємства. На підприємстві "Ontario Quality Motors" це програмне забезпечення активно використовується для обліку транспортних засобів, моніторингу їхнього стану та ефективного управління продажами. Завдяки Hillz Dealer менеджери мають змогу оперативно отримувати детальну інформацію про кожний автомобіль, включаючи дані про марку, модель, рік випуску, технічні характеристики, пробіг та історію власності. Це дозволяє покращити обслуговування клієнтів, оскільки вся необхідна інформація доступна в одному місці. Серед ключових переваг системи можна виділити її інтеграцію з

платформою Carfax, яка забезпечує прозорість процесу продажу, дозволяючи отримувати історичні дані про автомобілі, включаючи аварії та ремонти.

Проте існують і певні недоліки. Система не підтримує мобільний доступ, що ускладнює використання поза офісом. Крім того, Hillz Dealer не має функціоналу для управління записами клієнтів на тест-драйви чи моніторингу статусу підготовки автомобіля до видачі. Це створює додаткові ручні процеси, що уповільнюють роботу персоналу.

Отже, попри значний функціонал Hillz Dealer, підприємству доцільно розглянути впровадження додаткових рішень для усунення існуючих недоліків та оптимізації процесів.

DealerTrack є ще одним важливим програмним забезпеченням, яке використовується на підприємстві для управління фінансовими операціями, включаючи кредитування та оформлення угод. Ця система дозволяє автоматизувати процеси кредитної перевірки, забезпечує інтеграцію з банківськими установами та спрощує управління фінансовою документацією. DealerTrack також підтримує електронний документообіг, що значно скорочує час на підготовку та оформлення контрактів. Проте система має обмеження у сфері детального фінансового обліку, зокрема щодо автоматизованого моніторингу витрат, доходів та комісійних виплат. Це змушує працівників вручну збирати та узагальнювати фінансові дані, що може призводити до помилок і затримок у підготовці звітів для керівництва. DealerTrack також підтримує електронний документообіг, що значно скорочує час на підготовку та оформлення контрактів.

Окрім програмних систем, "Ontario Quality Motors" використовує корпоративний веб-сайт, який є важливим інструментом взаємодії з клієнтами. Веб-сайт надає можливість потенційним покупцям переглядати актуальний асортимент автомобілів, включаючи детальну інформацію про технічні характеристики, пробіг, рік випуску та ціну. Завдяки зручному інтерфейсу клієнти можуть відфільтрувати автомобілі за потрібними параметрами, що полегшує вибір.

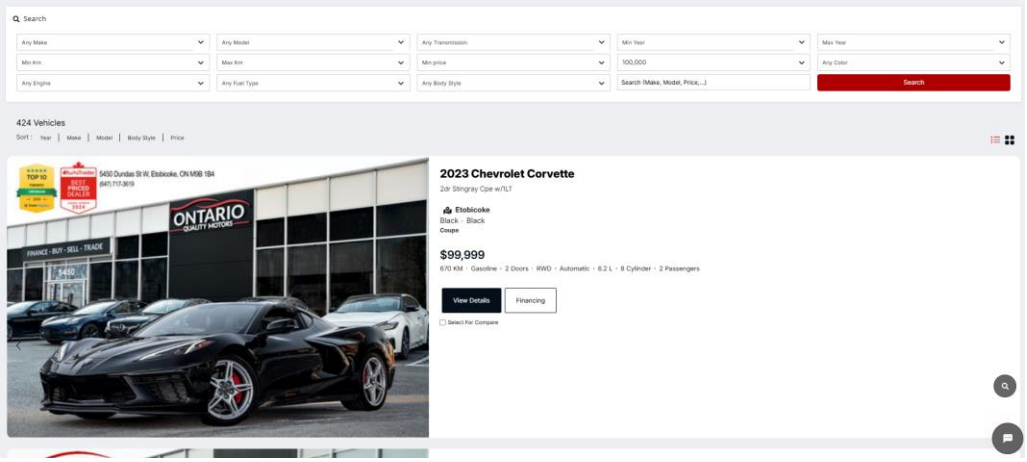


Рис. 1.3 Інтерфейс веб-сайту з вибором автомобілів[7]

Крім того, веб-сайт дозволяє зв’язатися з менеджерами для отримання додаткової інформації або запису на тест-драйв.

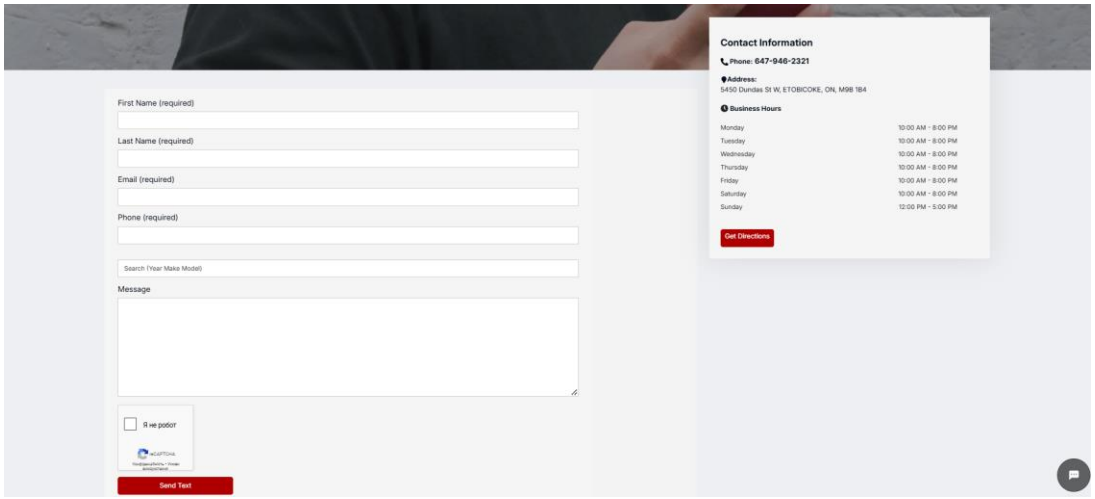


Рис. 1.4 Форма контактної заявки на веб-сайті Ontario Quality Motors[7]

Однак функціонал веб-сайту не включає можливості автоматизації запису клієнтів чи управління їх запитами, що ускладнює роботу рецепції.

Суттєвою проблемою підприємства є відсутність програмного забезпечення для автоматизації ключових адміністративних процесів, таких як управління записами клієнтів, моніторинг статусу продажу та підготовки автомобілів. Наприклад, клієнт, який бажає записатися на тест-драйв, повинен звернутися до рецепції, де працівник записує запит на папері та вручну передає інформацію

продавцю. Перед тим, як зробити запис, працівник рецепції змушений перевіряти доступність продавця, що спричиняє затримки та ускладнює процес обслуговування. Відсутність системи для централізованого управління цими запитами створює ризик втрати або некоректного виконання клієнтських запитів.

Крім того, недостатній рівень автоматизації в обліку продажів змушує продавців вручну відслідковувати, які автомобілі були реалізовані, і самостійно готувати звіти про кількість угод. Така практика значно ускладнює процес нарахування комісійних для власника підприємства та підвищує ризик помилок у фінансовому обліку. Ці недоліки не лише уповільнюють робочі процеси, але й можуть негативно впливати на задоволеність клієнтів та внутрішню ефективність роботи команди.

1.2 Світовий досвід впливу цифровізації на ефективність бізнес-процесів

Цифровізація є процесом впровадження цифрових технологій у всі аспекти діяльності підприємств, включаючи автоматизацію операцій, аналіз великих даних, інтеграцію інформаційних систем, розвиток електронної комерції та застосування штучного інтелекту. Вона створює нові можливості для підвищення ефективності, оптимізації ресурсів та забезпечення конкурентоспроможності бізнесу в умовах глобальної економіки, що динамічно змінюється.

Важливість цифровізації полягає у її здатності сприяти інноваціям, зменшувати витрати та покращувати взаємодію з клієнтами. Вона відкриває доступ до нових ринків, підвищує гнучкість компаній у реагуванні на виклики та забезпечує стійкість бізнесу під час криз, таких як пандемія COVID-19. Цифрові технології не лише трансформують внутрішні бізнес-процеси, але й змінюють способи взаємодії з клієнтами, постачальниками та партнерами, формуючи нові бізнес-моделі.

Північна Америка є провідним регіоном у глобальній цифровій трансформації. США та Канада мають потужну цифрову інфраструктуру, високий

рівень інновацій та широкий досвід інтеграції цифрових рішень у бізнес-процеси. Цей регіон демонструє, як використання сучасних технологій сприяє зростанню продуктивності, зниженню витрат і зміцненню конкурентних позицій на світовому ринку.

Цифровізація бізнесу стала невід'ємною частиною сучасної економіки США, демонструючи значні можливості для підвищення продуктивності, впровадження інновацій та адаптації до ринкових викликів. Одним із ключових аспектів цифрової трансформації є хмарні обчислення, які дозволяють підприємствам масштабувати свої IT-ресурси, зменшувати витрати та підвищувати гнучкість. Масштаби цифровізації в США є вражаючими, оскільки 90% підприємств здійснили цифровізацію хоча б однієї бізнес-функції. Найбільш поширеними напрямами цифровізації є фінансові та кадрові функції, які часто інтегруються одночасно.

Цифровізація бізнесу демонструє значну різноманітність підходів у різних галузях, що відображає специфіку їхньої діяльності, рівень технологічної інтеграції та доступність сучасних інструментів. Рисунок 1.5 ілюструє різноманітність підходів до цифровізації у різних галузях, відображаючи рівень технологічної інтеграції та використання хмарних обчислень. Інформаційний сектор має найвищі показники цифровізації та використання хмарних обчислень. Це пояснюється його безпосередньою залежністю від технологій, що забезпечують обробку великих обсягів даних, автоматизацію процесів і впровадження інноваційних рішень для аналізу. Даний сектор є прикладом того, як цифрові технології стають основою для досягнення конкурентних переваг і підвищення ефективності.

Професійні, наукові та технічні послуги також демонструють високі показники цифровізації та використання хмарних обчислень. Ці галузі активно інтегрують цифрові рішення для аналізу великих даних, автоматизації бізнес-процесів і розробки моделей прогнозування. Високий рівень цифровізації дозволяє їм бути одними з лідерів у впровадженні інноваційних підходів. Освітні послуги також займають провідні позиції за рівнем цифровізації та використання

хмарних обчислень. Це зумовлено широким використанням електронних платформ для управління навчальними процесами, впровадженням дистанційного навчання та використанням хмарних технологій для зберігання і обробки даних.

Водночас традиційні галузі, такі як сільське господарство, демонструють значно нижчі показники цифровізації та використання хмарних обчислень . Це відображає структурні виклики, такі як недостатній доступ до сучасних технологій у сільській місцевості, високі витрати на впровадження інноваційних рішень і низький рівень цифрової грамотності працівників. Попри це, сільське господарство поступово інтегрує новітні технології, такі як платформи для управління агробізнесом, що створює потенціал для його подальшого розвитку.

Роздрібна торгівля демонструє неоднорідні результати. Залежно від підгалузі, показники цифровізації варіюються в межах від 59.63% до 68.79%, а використання хмарних обчислень — від 34.55% до 43.43%. Роздрібна торгівля активно впроваджує електронну комерцію, онлайн-платформи для взаємодії з клієнтами та інструменти аналітики для персоналізації послуг, що забезпечує конкурентні переваги для великих гравців ринку. Однак менші підприємства стикаються з обмеженнями, зумовленими недостатніми ресурсами для масштабного впровадження цифрових рішень.

Аналіз демонструє значну різницю між галузями в контексті цифровізації. Сектори з високим рівнем залежності від технологій, такі як інформаційний, професійний і науковий, значно випереджають традиційні галузі, такі як сільське господарство. Рисунок 1.5 підкреслює, що інформаційний сектор є лідером цифровізації, тоді як сільське господарство залишається найбільш відстаючим. Ці диспропорції зумовлені як технологічними, так і економічними факторами, що вимагає спеціальних заходів для зменшення цифрового розриву між галузями. Зокрема, це може включати створення спеціальних програм підтримки для традиційних галузей, спрямованих на доступ до сучасних технологій, зниження витрат на впровадження та навчання працівників.[11]

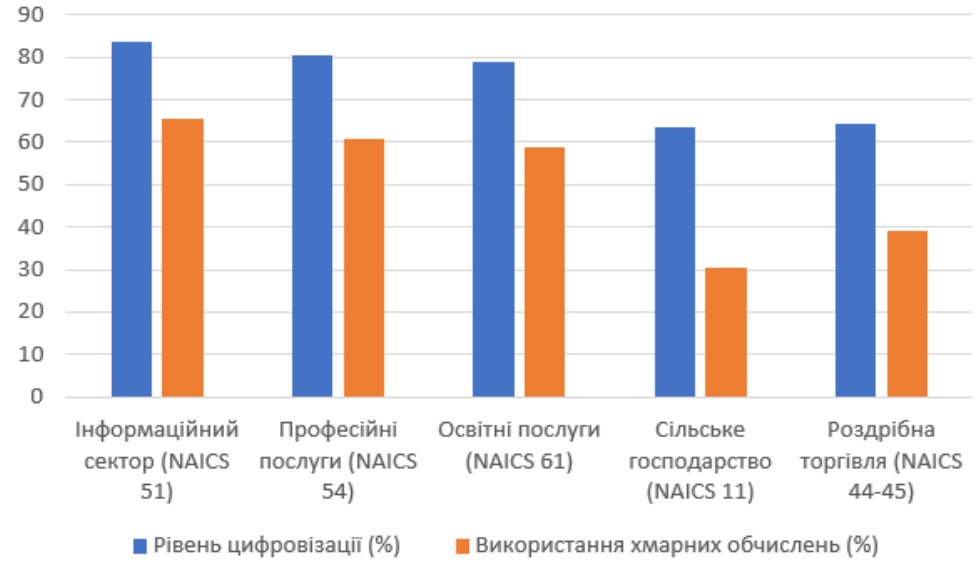


Рис. 1.5 Рівень цифровізації та використання хмарних обчислень у різних галузях [11]

Рівень впровадження хмарних обчислень та цифровізації значною мірою залежить від розміру підприємства, що відображає доступність ресурсів, управлінську гнучкість та стратегічні пріоритети компаній. Як свідчать дані рисунку 1.6, малі підприємства (1–4 працівників) демонструють найнижчі показники використання хмарних обчислень, які становлять лише 40.52%. Це обумовлено обмеженістю фінансових та технічних ресурсів, а також високими витратами на впровадження технологій, які ці підприємства часто не можуть дозволити. Водночас рівень загальної цифровізації у таких компаній становить 64.58%, що свідчить про певне використання базових цифрових рішень для виконання щоденних операцій.

Підприємства середнього розміру, особливо ті, які мають від 100 до 249 працівників, демонструють найвищі показники впровадження як цифровізації (86.58%), так і хмарних обчислень (63.21%). Це зумовлено активним використанням цифрових рішень для автоматизації бізнес-процесів, зменшення витрат та оптимізації операцій. Середні підприємства, як правило, володіють достатнім обсягом ресурсів для інвестування у цифровізацію, що дозволяє їм ефективно конкурувати на ринку та досягати стратегічних переваг.

Великі підприємства (понад 1000 працівників) демонструють середній рівень цифровізації, який становить 79.18%, та використання хмарних обчислень на рівні 54.62%. Незважаючи на високий ресурсний потенціал, великі компанії стикаються зі складнощами, пов’язаними зі структурними викликами, масштабуванням хмарних рішень та необхідністю інтеграції багатьох внутрішніх систем. Такі компанії використовують хмарні обчислення для оптимізації масштабних операцій, однак процес інтеграції ускладнюється необхідністю координації між підрозділами, забезпеченням безпеки даних та значними витратами на підтримку цифрової інфраструктури.

Загальний тренд свідчить про те, що зі зростанням розміру підприємства підвищується рівень впровадження цифрових технологій, проте до певного моменту. Найвищі показники спостерігаються у середніх компаній, які ефективно інтегрують інновації, поєднуючи управлінську гнучкість із достатнім обсягом ресурсів. Водночас малі підприємства потребують додаткової підтримки для подолання фінансових та технічних бар’єрів, а великі компанії потребують рішень, які б спрощували масштабування та інтеграцію хмарних технологій. Ці відмінності підкреслюють необхідність розробки спеціальних стратегій цифровізації для кожної категорії підприємств, орієнтованих на їхні унікальні потреби та виклики.[5]

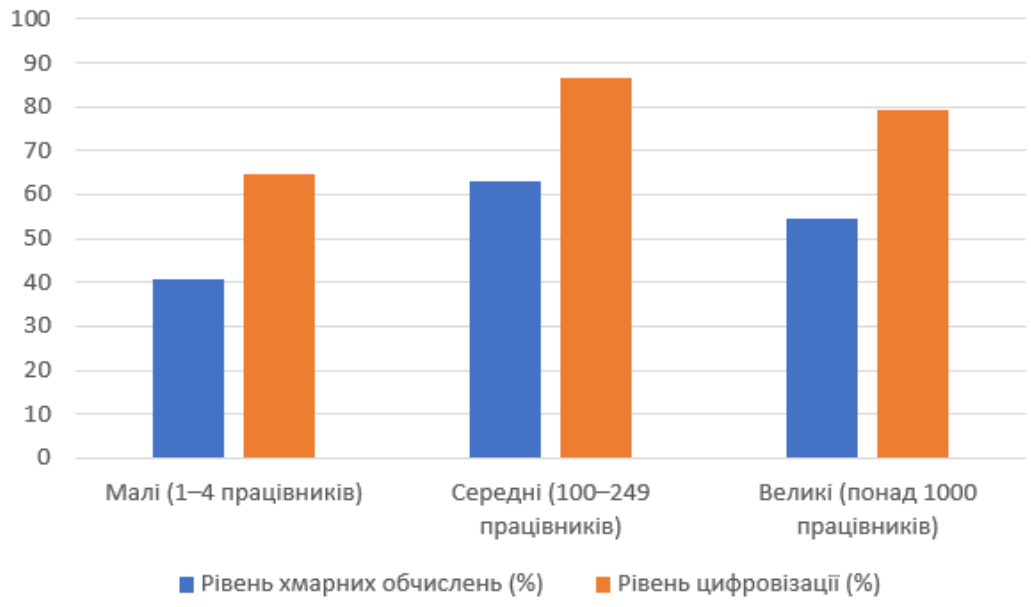


Рис. 1.6 Рівень цифровізації та використання хмарних обчислень за розміром підприємств [11]

Хмарні обчислення є важливим чинником стимулювання інновацій у бізнесі, особливо у сферах маркетингу, ділової практики та розробки продуктів. Дані рисунка 1.7 свідчать, що компанії, які активно використовують хмарні технології, демонструють значно вищий рівень впровадження інновацій порівняно з тими, які не застосовують ці рішення. Це підкреслює стратегічне значення хмарних обчислень у підвищенні ефективності та конкурентоспроможності сучасного бізнесу. Незважаючи на значний потенціал хмарних технологій, багато компаній стикаються з перешкодами на шляху до їх впровадження. Як видно з даних рисунка 1.7, компанії впроваджують різні види інновацій завдяки хмарним технологіям з наступною частотою:

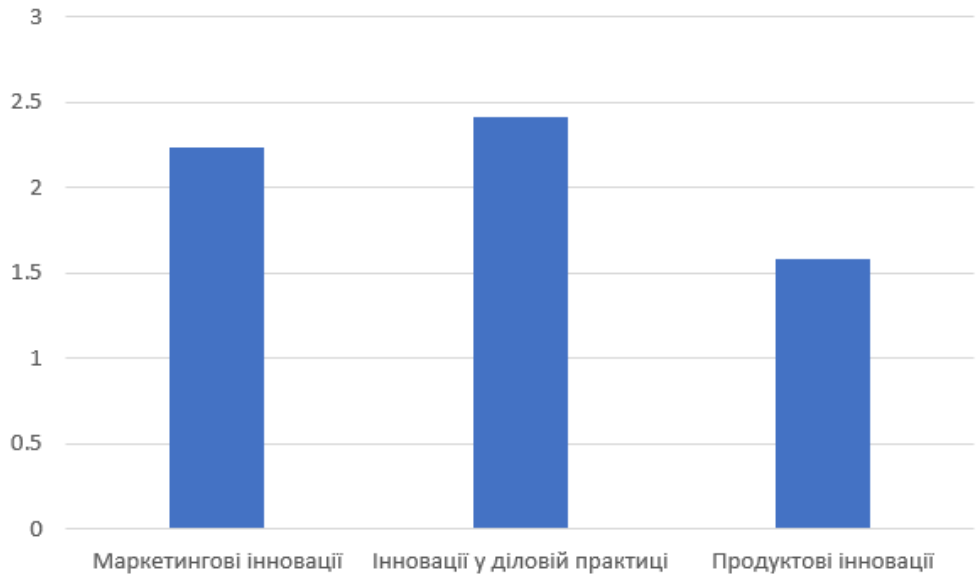


Рис. 1.7 Частота впровадження інновацій завдяки хмарним технологіям[5]

Серед основних бар’єрів на шляху впровадження хмарних технологій виділяються висока вартість (6.92%) та занепокоєння щодо безпеки даних (4.14%). Також 30.39% підприємств вважають, що хмарні технології не відповідають їхнім бізнес-потребам. Понад половина підприємств не стикаються

із серйозними перешкодами, що свідчить про готовність ринку до подальшого впровадження хмарних технологій. Основні бар'єри впровадження відображені у рисунку 1.8:

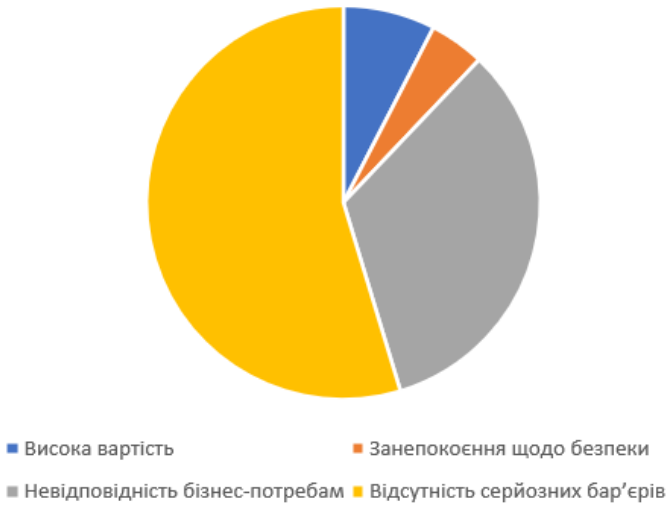


Рис. 1.8 Основні бар'єри впровадження хмарних технологій у бізнесі[5]

Цифровізація бізнесу в США демонструє високий рівень інтеграції цифрових технологій, зокрема хмарних обчислень, які сприяють інноваціям та підвищенню ефективності. Галузі, орієнтовані на інформаційні технології та науки, залишаються лідерами у впровадженні цифрових рішень, тоді як традиційні сектори та малі підприємства потребують додаткової підтримки для зменшення цифрового розриву. Хмарні обчислення відіграють ключову роль у трансформації бізнесу, забезпечуючи гнучкість, ефективність та доступ до нових інструментів для інновацій. Подальше впровадження цих технологій, особливо серед малих підприємств, може суттєво підвищити конкурентоспроможність та інноваційний потенціал американської економіки.

Найбільший вплив хмарних технологій спостерігається у маркетингових інноваціях. Компанії, що використовують хмарні обчислення, впроваджують маркетингові інновації у 2.24 рази частіше, ніж ті, які не використовують ці технології. Такий результат зумовлений доступом до передових інструментів аналізу ринкових даних, автоматизації рекламних кампаній і персоналізації

пропозицій для клієнтів. Хмарні рішення дозволяють компаніям ефективніше працювати з великими обсягами інформації, підвищуючи якість прийняття маркетингових рішень.

Інновації у діловій практиці також демонструють значний зв'язок із хмарними технологіями. Компанії-користувачі хмарних рішень у 2.41 рази частіше впроваджують такі інновації, ніж ті, що не інтегрують хмару у свої бізнес-процеси. Цей вплив обумовлений можливістю автоматизації ключових внутрішніх операцій, таких як документообіг, управління персоналом і адміністрування. Завдяки використанню хмарних платформ компанії можуть знизити витрати часу та ресурсів на виконання рутинних завдань, підвищуючи загальну продуктивність.

У продуктових інноваціях вплив хмарних обчислень менш виражений, але все ще суттєвий. Дані рисунка показують, що компанії, які використовують хмарні технології, впроваджують продуктові інновації у 1.58 рази частіше, ніж ті, хто не користується цими технологіями. Це пояснюється спрощеним доступом до аналітики, кращою координацією між різними підрозділами компанії та інтеграцією зовнішніх ресурсів, що дозволяє створювати нові продукти більш ефективно.

Таким чином, дані рисунків 1.7 та 1.8 ілюструють, що хмарні обчислення є важливим інструментом для розвитку інновацій у бізнесі. Компанії, які інтегрують ці технології, отримують значні переваги у маркетинговій діяльності, оптимізації внутрішніх процесів та розробці нових продуктів. Це підтверджує важливість хмарних рішень для досягнення стратегічних цілей і довгострокового зростання бізнесу.[2]

Цифровізація також стала важливим чинником економічного розвитку Канади, особливо в умовах кризи COVID-19, коли впровадження цифрових технологій значно прискорилося. Протягом останніх двох десятиліть значна частина бізнесів активно інтегрувала цифрові рішення у свої виробничі процеси, що сприяло підвищенню продуктивності, адаптивності та гнучкості. У період пандемії використання цифрових інструментів стало вирішальним фактором для

підтримки бізнес-операцій, забезпечення дистанційної роботи, а також збереження економічної активності на рівні споживачів та підприємств.

Дослідження демонструють, що економічні результати секторів, які інтенсивно використовують цифрові технології, значно перевищують показники нецифрових секторів. Використання композитного індексу цифрової інтенсивності дозволяє оцінити рівень впровадження цифрових технологій у різних галузях економіки. Галузі з високою цифровою інтенсивністю включають виробництво транспортного обладнання, комп'ютерної та електронної техніки, фінансові послуги, страхування, а також професійні та технічні послуги.

Протягом останніх двох десятиліть продуктивність праці у цифрово інтенсивних секторах економіки зростала значно швидше, ніж у нецифрових секторах, що свідчить про важливість цифрових технологій для підвищення ефективності. Як показано на рисунку 1.10, продуктивність у цифрово інтенсивних секторах із 2002 по 2019 рік зросла на 22.1%, тоді як у нецифрових секторах цей показник становив лише 6.3%. [6]

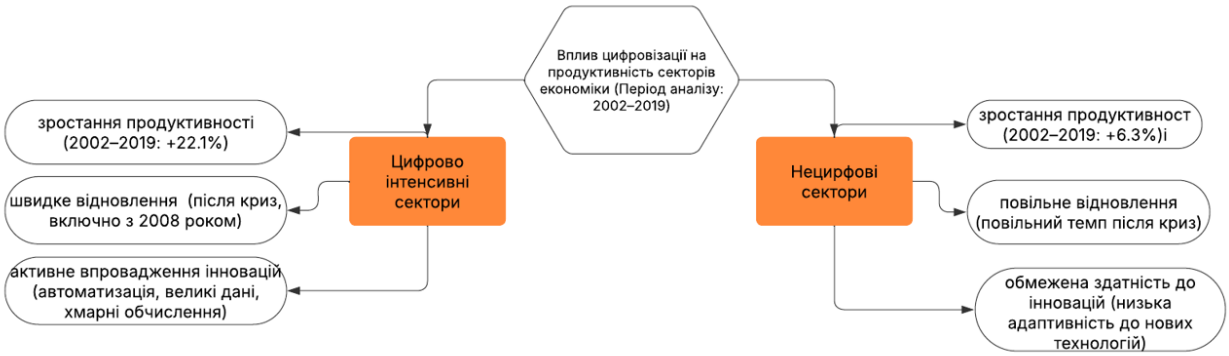


Рис. 1.9 Вплив цифровізації на продуктивність цифрово інтенсивних та нецифрових секторів економіки [6]

Аналіз рисунку 1.9 свідчить, що цифрові технології є важливим фактором економічного зростання, особливо для секторів, які активно впроваджують інноваційні рішення. Цифрово інтенсивні сектори демонструють більшу адаптивність до змін і здатність швидко відновлюватися після криз, що

підкреслює їхню роль у забезпеченні сталого економічного розвитку. Натомість повільне зростання продуктивності у нецифрових секторах вказує на необхідність активізації процесів цифровізації для скорочення розриву між секторами та підвищення їхньої конкурентоспроможності.

Пандемія COVID-19 суттєво вплинула на всі сектори економіки, однак цифрово інтенсивні сектори виявилися значно стійкішими до кризи порівняно з нецифровими, що демонструє рисунок 1.10. На рисунку відображено порівняння змін зайнятості та валового внутрішнього продукту (ВВП) у 2020 році для цифрово інтенсивних і нецифрових секторів.[6]

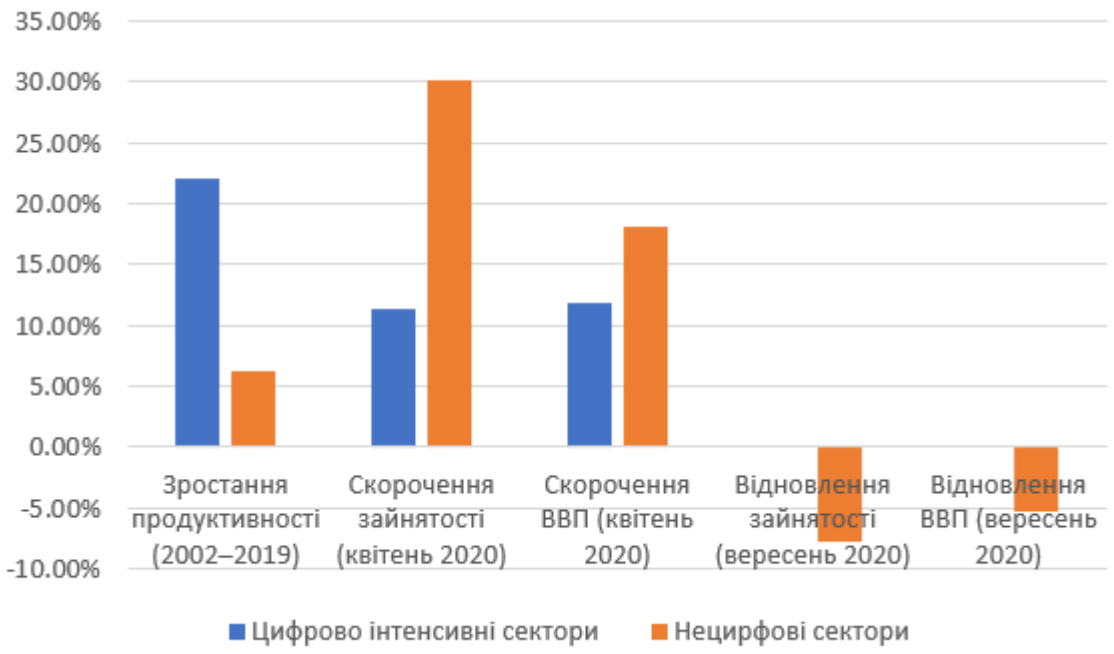


Рис. 1.10 Вплив COVID-19 на зайнятість і ВВП у цифрово інтенсивних та нецифрових секторах економіки [6]

У квітні 2020 року зайнятість у цифрово інтенсивних секторах скоротилася на 11.3%, тоді як у нецифрових секторах цей показник сягав 30.2%. Подібна тенденція спостерігалася у динаміці ВВП: скорочення в цифрових секторах становило 11.8%, а в нецифрових – 18.1%. До вересня 2020 року зайнятість у цифрових секторах досягла рівня 2019 року, що свідчить про їхню здатність до швидкої адаптації. Натомість у нецифрових секторах зайнятість залишалася на

7.8% нижчою за рівень 2019 року, а ВВП – на 5.3% нижчим. Ця різниця підкреслює переваги цифрових технологій у кризових умовах. [6]

Цифровізація в Канаді та США демонструє значний вплив на продуктивність праці, стійкість бізнесу до криз та здатність секторів економіки адаптуватися до нових викликів, таких як пандемія COVID-19. Активне впровадження цифрових технологій, особливо хмарних обчислень, значно покращує ефективність, інноваційність та конкурентоспроможність секторів, що підкреслює необхідність подальшої цифровізації для сталого розвитку економіки обох країн.

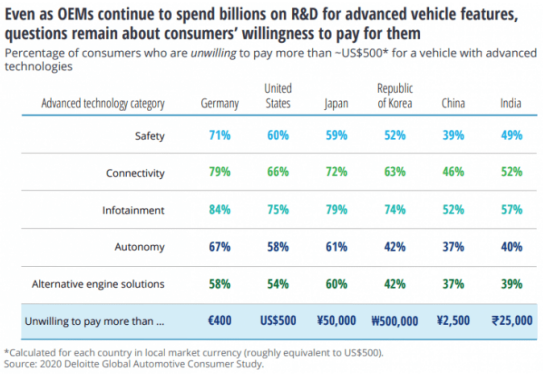
1.3 Виклики та технічні особливості впровадження цифрових технологій у бізнес, пов'язаний з обслуговуванням автомобілів

Автомобільна промисловість сьогодні переживає масштабну цифрову трансформацію, яка впливає на всі аспекти її функціонування, починаючи від виробництва та продажів і закінчуючи обслуговуванням клієнтів. Основними рушіями змін є Інтернет речей (IoT), підключені транспортні засоби, екологічні вимоги та зростаючі очікування клієнтів. Галузь активно інтегрує новітні технології для підвищення ефективності, адаптивності та задоволеності клієнтів, однак цей процес супроводжується як значними перевагами, так і численними викликами.

Однією з основних переваг цифровізації є підвищення ефективності виробничих процесів. Завдяки впровадженню IoT, наприклад, на заводі BMW у Регенсбурзі час запуску нових програм скоротився на 80%, а кількість дефектів зменшилася на 5%. Ця технологія забезпечує прогнозування несправностей обладнання та оптимізацію операцій у реальному часі. Такі інновації також дозволяють створювати розумні ланцюги постачання, які забезпечують кращу прозорість і синхронізацію процесів

Цифровізація також сприяє розвитку нових бізнес-моделей, таких як мобільність як послуга (MaaS), інтеграція прогнозного обслуговування та

підключених транспортних засобів. Tesla, наприклад, збирає дані з бортових датчиків, що дозволяє покращувати функціональність автопілота через бездротові оновлення програмного забезпечення. Водночас Volvo Polestar реалізує свої автомобілі виключно через онлайн-платформи, поєднуючи це з фізичними виставковими просторами, що створює унікальний досвід для клієнтів.



Even as OEMs continue to spend billions on R&D for advanced vehicle features, questions remain about consumers' willingness to pay for them

Percentage of consumers who are *unwilling* to pay more than ~US$500* for a vehicle with advanced technologies

Advanced technology category	Germany	United States	Japan	Republic of Korea	China	India
Safety	71%	60%	59%	52%	39%	49%
Connectivity	79%	66%	72%	63%	46%	52%
Infotainment	84%	75%	79%	74%	52%	57%
Autonomy	67%	58%	61%	42%	37%	40%
Alternative engine solutions	58%	54%	60%	42%	37%	39%
Unwilling to pay more than ...	€400	US$500	¥50,000	₩500,000	¥2,500	₹25,000

*Calculated for each country in local market currency (roughly equivalent to US$500).
Source: 2020 Deloitte Global Automotive Consumer Study.

Рис. 1.11 Готовність споживачів у різних країнах платити за вдосконалені автомобільні технології (2020 рік) [5]

На рисунку 1.11 , відображає рівень готовності клієнтів до оплати інноваційних функцій, помітно, що більшість споживачів у Німеччині та Японії не готові платити понад 500 доларів за автономні системи або альтернативні двигуни. Це підкреслює, що попит на деякі інновації залишається обмеженим, що ускладнює комерціалізацію таких технологій.

Водночас одним із викликів є висока вартість впровадження цифрових рішень. Розробка та інтеграція IoT, AI і підключених транспортних засобів вимагає значних інвестицій, що може бути перешкодою для малих і середніх компаній. Крім того, фрагментація інфраструктури зарядних станцій для електромобілів залишається серйозною проблемою. Незважаючи на те, що провідні виробники, такі як Volkswagen, прогнозують продажі 1.4 мільйона електромобілів до 2025 року, недостатня інфраструктура гальмує їх масове впровадження.

Важливим напрямом цифровізації є впровадження технологій доповненої реальності (AR) та віртуальної реальності (VR). Наприклад, Volkswagen використовує AR-додатки для технічного обслуговування, що дозволяє скорочувати час на ремонт і підвищувати точність виконання робіт. Віртуальні шоуруми також змінюють досвід клієнтів, дозволяючи їм випробувати автомобіль у симуляторі без необхідності відвідувати фізичні дилерські центри.

На рисунку 1.12 , ілюструє розвиток електромобілів та їхню інтеграцію в цифрові платформи, показано, як продажі електромобілів зростають завдяки цифровій комерції. Наприклад, Amazon замовила 1800 електрофургонів у Mercedes, що демонструє популярність екологічно чистих транспортних засобів серед великих компаній.

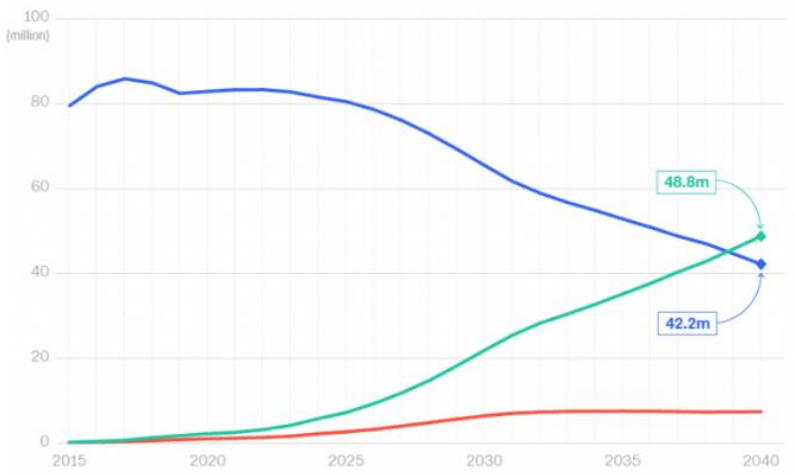


Рис. 1.12 Прогноз продажів електромобілів, гібридів та автомобілів із двигунами внутрішнього згоряння до 2040 року [5]

Рисунок 1.12 демонструє прогнозовану динаміку продажів автомобілів із двигунами внутрішнього згоряння, електромобілів і гібридів до 2040 року. Продажі електромобілів (зелена лінія) стрімко зростають і перевищать продажі автомобілів із двигунами внутрішнього згоряння (синя лінія) до 2040 року, тоді як гібриди (червона лінія) демонструють повільне зростання. Ця тенденція свідчить про перехід до електрифікації автомобільного ринку, зумовлений екологічними вимогами, інноваціями та зміною споживчих пріоритетів. [5]

Цифровізація в автомобільній промисловості відкриває численні можливості для оптимізації процесів, підвищення ефективності та покращення клієнтського досвіду. Водночас галузь стикається з викликами, такими як високі інвестиційні бар'єри, опір клієнтів і недостатня цифрова зрілість. Успіх трансформації залежить від здатності компаній інтегрувати нові технології поступово, зосереджуючись на довгостроковій рентабельності та стратегічному плануванні. [1]

1.4 Прогнозування підвищення ефективності управління процесами після впровадження мобільного додатку для Ontario Quality Motors

Автомобільний салон "Ontario Quality Motors" функціонує в умовах жорсткої конкуренції, що створює потребу в постійній оптимізації бізнес-процесів та вдосконаленні взаємодії з клієнтами. Одним із перспективних напрямів розвитку компанії є впровадження мобільного додатку, який забезпечить автоматизацію ключових операцій, підвищить ефективність роботи салону та посилить його конкурентоспроможність.

Як свідчить аналіз ключових функцій мобільного додатку (рис. 1.13), впровадження цієї технології сприятиме централізованому управлінню клієнтською базою, включаючи збереження контактної інформації, історії звернень, записів на тест-драйви та бронювань автомобілів. Очікується, що автоматизація цих процесів дозволить скоротити час обробки клієнтських запитів на 40% у перші два роки, що, своєю чергою, підвищить задоволеність клієнтів. Інтеграція онлайн-чату для оперативного консультування дозволить знизити кількість пропущених звернень на 30% та підвищити лояльність клієнтів на 20% завдяки швидкому реагуванню на їхні запити.

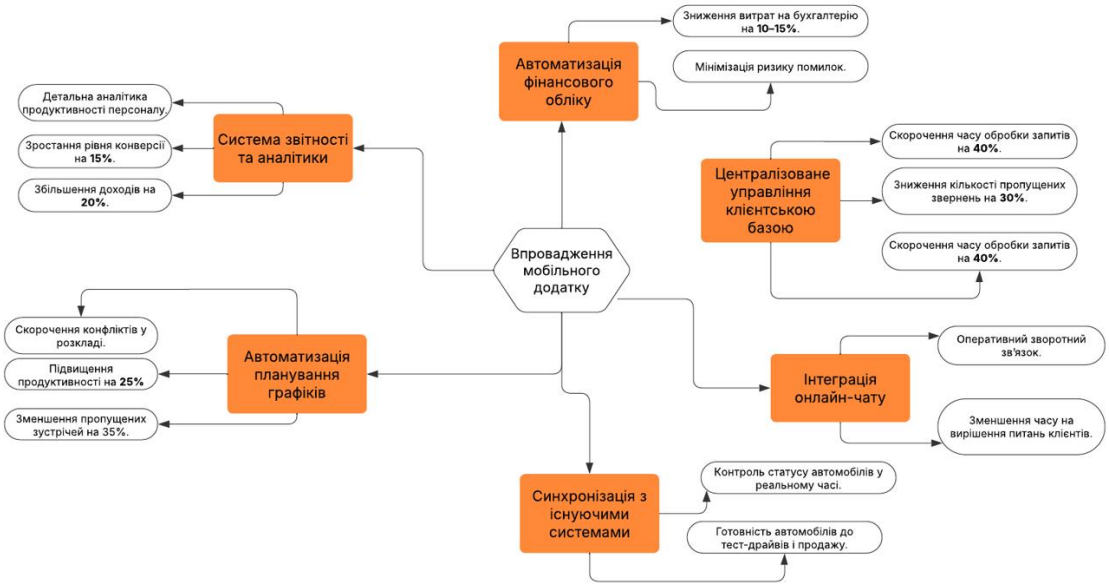


Рис. 1.13 - Ключові функції та результати впровадження мобільного додатку в автомобільному салоні

Додаток також забезпечить синхронізацію з інформаційними системами Hillz Dealer та DealerTrack, що дозволить співробітникам салону у реальному часі контролювати статус автомобілів, їхню готовність до продажу та тест-драйвів. У середньостроковій перспективі (3–4 роки) це дозволить зменшити адміністративне навантаження на персонал на 50% завдяки автоматизації процесів запису та моніторингу підготовки автомобілів. Крім того, система автоматизованого планування графіків роботи допоможе уникати конфліктів у розкладі, підвищуючи продуктивність персоналу на 25%, а функція автоматичних нагадувань про заплановані зустрічі знизить кількість пропущених візитів на 35%.

Іншим важливим аспектом впровадження мобільного додатку є доступ керівництва до детальної аналітики щодо продажів, продуктивності співробітників та ефективності маркетингових кампаній. Ці дані дозволять компанії розробляти персоналізовані маркетингові стратегії, що, за прогнозами, забезпечить підвищення рівня конверсії на 15% та зростання загального доходу на 20% упродовж п’яти років. Крім того, автоматизація фінансового обліку дозволить скоротити обсяг ручного введення даних, зменшити ризик помилок і

знизити витрати на бухгалтерський супровід на 10–15% у перші два роки використання додатку.

Очікується, що на п’ятому році функціонування мобільного додатку його функціонал буде розширено за допомогою впровадження технологій штучного інтелекту для аналізу поведінки клієнтів та прогнозування попиту. Це дозволить покращити персоналізацію послуг, підвищити середній чек покупки та забезпечити компанії більшу адаптивність до змін ринку. Інтеграція мобільного додатку з іншими системами компанії створить єдину цифрову екосистему, яка забезпечить зручність використання та підвищить ефективність роботи підприємства.

Рисунок 1.14 ілюструє очікувані результати впровадження мобільного додатку, серед яких найважливіші: скорочення часу обробки клієнтських запитів на 40%, зниження кількості пропущених звернень на 30%, підвищення лояльності клієнтів на 20%, зменшення адміністративного навантаження на 50% та підвищення продуктивності персоналу на 25%. Крім того, зниження витрат на бухгалтерський супровід на 10–15% та зростання доходів компанії на 20% свідчать про значний фінансовий ефект від впровадження додатку.

Рис. 1.14 Прогноз Впливу мобільного додатку на ефективність бізнес-процесів та клієнтський сервіс

Впровадження мобільного додатку стане важливим етапом цифрової трансформації салону "Ontario Quality Motors", сприяючи зниженню адміністративного навантаження, покращенню обслуговування клієнтів та підвищенню загальної ефективності роботи. У довгостроковій перспективі мобільний додаток забезпечить стабільний розвиток компанії, дозволяючи їй адаптуватися до змін ринку та залишатися конкурентоспроможною.[1]

РОЗДІЛ 2. Технологічні рішення для оптимізації бізнес-процесів

2.1. Огляд та аналіз ефективності існуючих рішень управління процесами підприємств

Ефективне управління бізнес-процесами є критично важливим аспектом для забезпечення конкурентоспроможності підприємства на сучасному ринку. Воно дозволяє не тільки оптимізувати внутрішні операції, а й підвищити продуктивність персоналу, зменшити витрати, поліпшити якість обслуговування клієнтів та забезпечити гнучкість у прийнятті рішень. В умовах швидкого розвитку технологій, автоматизація бізнес-процесів за допомогою цифрових інструментів стала необхідною для будь-якого підприємства, яке прагне досягти високих результатів у своїй діяльності. У цьому контексті компанія Ontario Quality Motors активно використовує кілька сучасних інструментів для управління своїми бізнес-процесами, що дозволяє ефективно взаємодіяти з клієнтами, оптимізувати операції з продажу та фінансування, а також забезпечити контроль над технічним станом автомобілів.

Для управління бізнес-процесами компанія використовує три основні інструменти: Hillz Dealer, DealerTrack та корпоративний веб-сайт. Кожен з цих інструментів має свою специфіку та виконує важливі функції в організації роботи автосалону.

Компанія *Ontario Quality Motors* використовує систему **Hillz Dealer** як ключовий інструмент автоматизації обліку автомобілів, управління процесами

продажу та контролю за сервісним обслуговуванням. Проведений аналіз ефективності функціонування цієї системи свідчить про наявність суттєвих переваг, проте порівняння з провідними світовими аналогами — такими як **DealerSocket** та **AutoRaptor** — виявляє низку обмежень, що можуть впливати на загальну продуктивність управлінських процесів.

Hillz Dealer демонструє високий рівень автоматизації обліку транспортних засобів, забезпечуючи зберігання повної інформації щодо технічних характеристик, історії обслуговування, пробігу та поточного стану автомобіля. Однією з ключових переваг є інтеграція з платформою **Carfax**, що підвищує прозорість і довіру клієнтів при здійсненні угод. Оперативний доступ до інформації, що оцінюється на рівні **85%**, дає змогу менеджерам швидко обробляти запити клієнтів, а автоматизований супровід процесу продажу (**80%**) сприяє підвищенню ефективності взаємодії з покупцями. Крім того, цифрове введення даних дозволяє зменшити ризик помилок у фінансовій та технічній документації (**75%**).

Утім, порівняння з хмарною платформою **DealerSocket**, яка активно використовується великими дилерськими мережами у США, засвідчує функціональні обмеження системи Hillz Dealer. DealerSocket забезпечує повноцінний **мобільний доступ** до всіх ключових функцій: перегляд інформації про транспортні засоби, управління лідами, онлайн-бронювання тест-драйвів та електронну комунікацію з клієнтами. У той час, як Hillz Dealer не підтримує мобільні інтерфейси, що істотно знижує гнучкість персоналу у віддаленій роботі. Також система не має функції автоматизованого бронювання тест-драйвів, що змушує менеджерів здійснювати цей процес вручну, з ефективністю лише **45%**. Це створює додаткове навантаження на персонал і уповільнює обслуговування клієнтів.

Інформаційна система **AutoRaptor**, орієнтована переважно на малі та середні автосалони, також демонструє переваги в контексті зручності та адаптивності. Інтуїтивно зрозумілий інтерфейс, підтримка мобільних платформ і гнучкі налаштування шаблонів комунікації роблять її привабливою

альтернативою для компаній, орієнтованих на ефективну взаємодію з клієнтами. У цьому контексті Hillz Dealer поступається за рівнем клієнтського сервісу та функціональністю в частині управління сервісними процесами.

Ще однією суттєвою відмінністю між розглянутими системами є **рівень аналітичної підтримки**. DealerSocket пропонує користувачам розширені модулі звітності, інструменти фінансового прогнозування та засоби підтримки стратегічного планування. Натомість Hillz Dealer обмежується базовим функціоналом звітності, що ускладнює довгострокове планування, аналітику ефективності продажів та ухвалення управлінських рішень на основі даних.

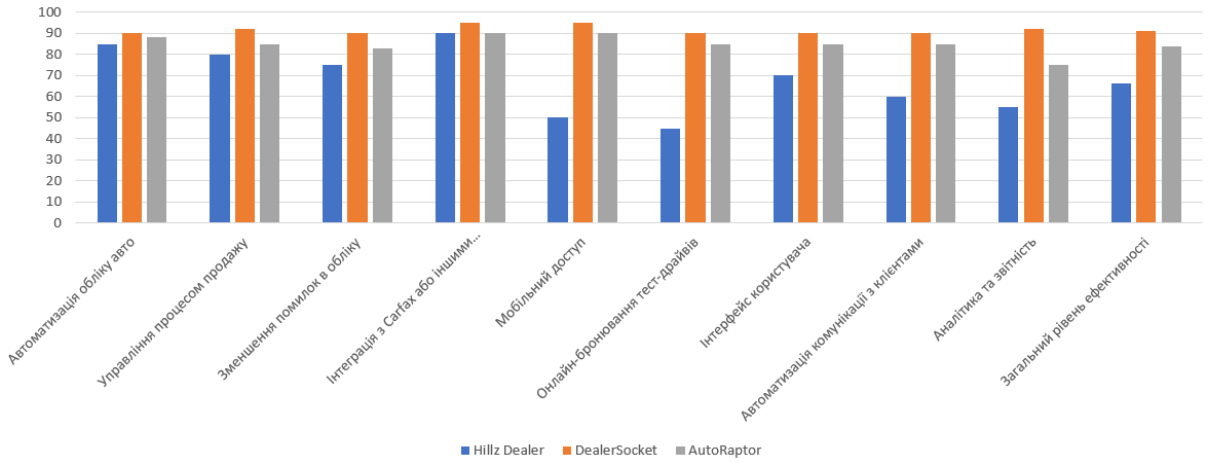


Рис. 2.1 Порівняльна оцінка ефективності DMS-систем Hillz Dealer, DealerSocket та AutoRaptor

На рисунку 2.1 представлено **порівняльну оцінку трьох інформаційних систем**, що використовуються в автосалонах для управління бізнес-процесами: Hillz Dealer, DealerSocket та AutoRaptor. Оцінювання здійснено за десятьма ключовими критеріями, серед яких: автоматизація обліку транспортних засобів, управління процесами продажу, точність даних, інтеграція з Carfax, мобільний доступ, онлайн-бронювання, зручність інтерфейсу, автоматизація комунікації, аналітика та загальний рівень ефективності.

Як видно з діаграми, **DealerSocket** демонструє найвищі показники у більшості категорій, зокрема у мобільному доступі (**95%**), аналітичних функціях

(**92%**) та автоматизованій взаємодії з клієнтами (**90%**). **AutoRaptor** посідає проміжне місце, поєднуючи зручний інтерфейс і достатній рівень автоматизації базових процесів. У свою чергу, **Hillz Dealer** показує стабільно високі результати у таких сферах, як облік авто (**85%**) і продажі (**80%**), проте поступається конкурентам за функціональністю мобільного доступу (**50%**) та інтеграції онлайн-сервісів (**45%**).

Інформаційна система **DealerTrack** є ключовим інструментом в організації фінансових процесів компанії *Ontario Quality Motors*. Вона активно використовується для автоматизації обробки кредитних заявок, зменшення кількості помилок у фінансовій документації, а також для взаємодії з фінансовими установами. Однією з її головних переваг є **автоматизована перевірка фінансової історії клієнтів**, яка дає змогу менеджерам оперативно отримувати достовірну інформацію про платоспроможність покупців та ухвалювати обґрунтовані кредитні рішення. За цим показником DealerTrack демонструє високий рівень ефективності — **85%**.

Крім того, система позитивно зарекомендувала себе у **зменшенні кількості помилок у фінансових документах**. Завдяки цифровому введенню даних та автоматизованим перевіркам, вона мінімізує ризики, пов’язані з людським фактором. Це дозволяє зменшити кількість повторних перевірок, підвищити достовірність кредитних угод та прискорити їх обробку. Ефективність за цим критерієм становить **80%**.

Однак, попри високі показники в частині оперативних функцій, **DealerTrack має певні функціональні обмеження**, що ускладнюють її використання для стратегічного фінансового управління. Зокрема, система не забезпечує повноцінного **автоматизованого моніторингу витрат і доходів**, що змушує бухгалтерів вручну вводити частину інформації у звітність. Це створює додаткове навантаження на персонал та підвищує ймовірність помилок. За цим показником ефективність системи оцінюється лише на **50%**.

Крім того, DealerTrack має **обмежені можливості фінансового прогнозування** — у системі відсутні інструменти для моделювання доходів,

аналізу фінансових ризиків або розрахунку сценаріїв витрат. Це суттєво обмежує застосування платформи в контексті стратегічного планування. Ефективність у сфері прогнозування становить **45%**, що вказує на потребу в інтеграції з додатковими аналітичними модулями.

Для більш повного розуміння переваг і обмежень DealerTrack доцільним є **порівняння з альтернативною системою RouteOne**, яка також широко використовується автодилерами у Північній Америці. Обидві системи мають високі показники продуктивності в частині обробки заявок та зменшення помилок, однак RouteOne перевершує DealerTrack за **аналітичними можливостями**. Зокрема, RouteOne має розширені інструменти **фінансового прогнозування (80%)**, **автоматизований моніторинг витрат (75%)**, а також дозволяє створювати гнучкі сценарії кредитування (**82%**) і інтегруватися з CRM-системами (**90%**), що значно розширює функціонал управління фінансами.

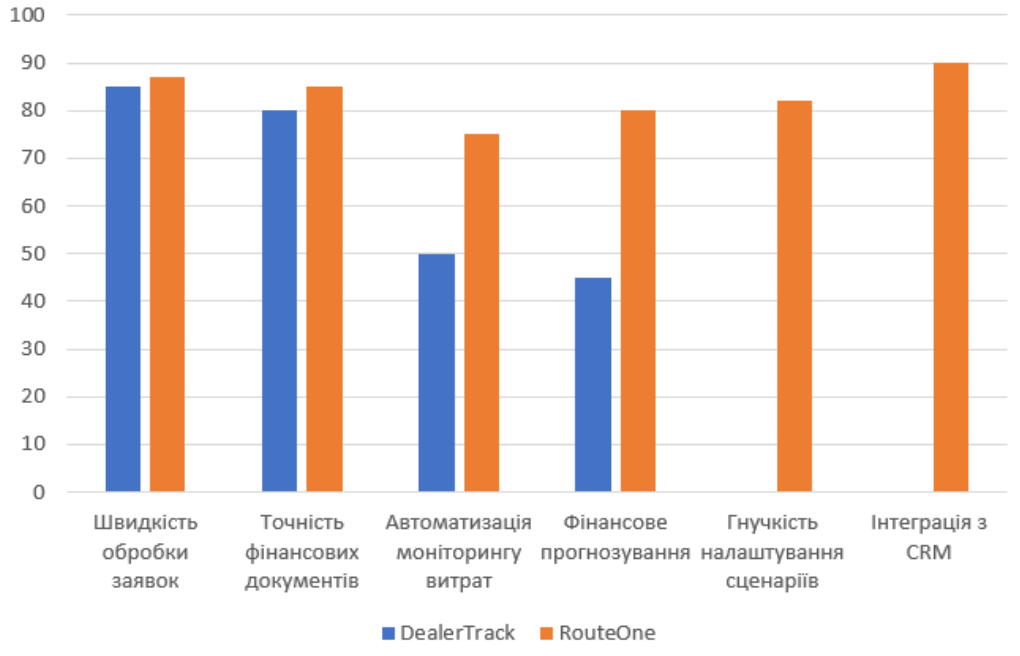


Рис. 2.2 Порівняння ефективності DealerTrack та RouteOne за ключовими фінансовими критеріями

На **рисунку 2.2** наведено порівняльну стовпчасту діаграму, яка відображає ефективність двох фінансових систем — DealerTrack та RouteOne — за шістьма

ключовими критеріями: **швидкість обробки заявок**, **точність фінансових документів**, **автоматизація моніторингу витрат**, **фінансове прогнозування**, **гнучкість налаштування кредитних сценаріїв** та **інтеграція з CRM**. Як видно з діаграми, DealerTrack демонструє стабільну ефективність у базових операційних процесах (85–80%), тоді як RouteOne забезпечує ширші можливості в частині аналітики, адаптивності та стратегічного управління. Це підкреслює потенціал RouteOne як більш гнучкого та сучасного інструменту для автодилерів, орієнтованих на довгострокове планування та ефективну фінансову аналітику.

Корпоративний вебсайт компанії *Ontario Quality Motors* є важливим інструментом презентації асортименту автомобілів, а також каналом взаємодії з потенційними клієнтами. Однією з ключових переваг ресурсу є **висока доступність інформації**. Завдяки зручному онлайн-каталогу, користувачі мають змогу швидко ознайомитися з технічними характеристиками, історією використання та вартістю транспортних засобів без необхідності фізичного відвідування автосалону. Це не лише економить час клієнтів, а й значно **зменшує навантаження на персонал**, оскільки багато базових запитів вирішуються ще до безпосереднього контакту з менеджером. За критерієм доступності вебсайт оцінюється на **90%**, що вказує на його ефективність як цифрового інструменту презентації продукції.

Окрім інформативності, вебсайт також виконує функцію **комунікаційного каналу**. Користувачі можуть залишати запити на зворотній зв’язок, ставити питання щодо автомобілів, уточнювати умови покупки або замовляти консультацію. Менеджери оперативно опрацьовують звернення, що позитивно впливає на **загальний рівень задоволеності клієнтів**. За цим показником вебсайт демонструє ефективність на рівні **80%**.

Втім, вебресурс має **функціональні обмеження**, які стримують його подальший розвиток. Одним із основних недоліків є **низький рівень автоматизації внутрішніх процесів**, зокрема обробки запитів на тест-драйви. Всі заявки надходять у вигляді форм зворотного зв’язку і обробляються вручну,

що створює потенційні затримки у відповідях і підвищує навантаження на персонал. Ефективність цього аспекту оцінюється на **40%**, що вказує на доцільність впровадження автоматизованих сервісів обробки запитів.

Ще одним обмеженням є **відсутність онлайн-системи бронювання тест-драйвів**. Клієнт не має змоги самостійно вибрати дату і час поїздки в інтерактивному режимі; замість цього він подає запит, після чого з ним зв'язується менеджер. Така модель уповільнює процес взаємодії, особливо в пікові години. Функціональність онлайн-бронювання оцінюється лише на **50%**, що свідчить про необхідність модернізації цього сегмента платформи.

У контексті конкурентного порівняння варто відзначити, що **багато сучасних автосалонів у Північній Америці вже впровадили повноцінні онлайн-платформи**, які підтримують **інтерактивне бронювання**, **автоматизовану розсилку підтверджень**, **чат-боти для первинної консультації**, а також **адаптацію під мобільні пристрої**. Наприклад, вебсайти компаній **AutoPlanet** та **CarNation Canada** дозволяють користувачам самостійно обирати модель авто, доступну годину тест-драйву та миттєво отримувати підтвердження. У порівнянні з цими платформами, вебсайт *Ontario Quality Motors* виглядає менш автоматизованим і менш гнучким у контексті самостійного керування процесом взаємодії.

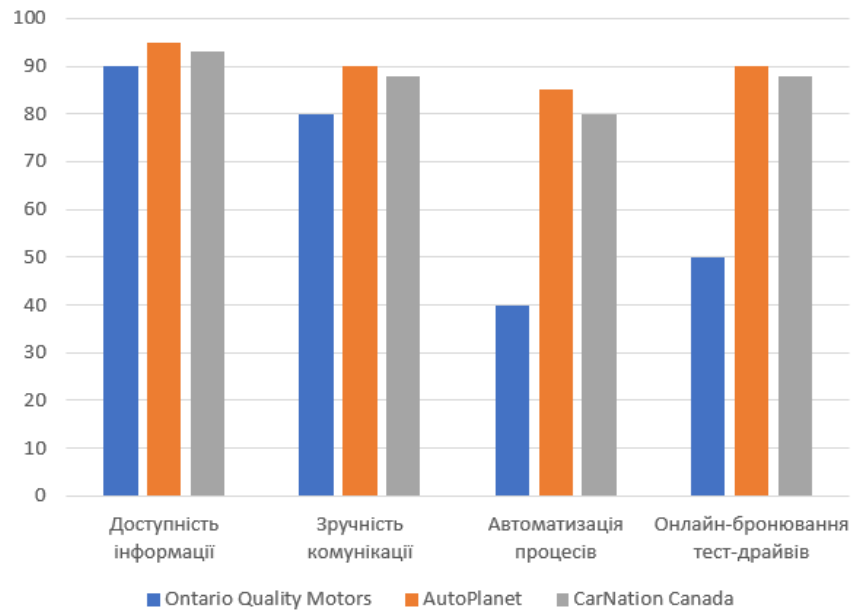


Рис. 2.3 Порівняльна оцінка функціональності корпоративного вебсайту Ontario Quality Motors

На рисунку 2.3 представлено діаграму, яка відображає рівень ефективності вебсайту компанії за чотирма ключовими критеріями: доступність інформації (90%), зручність комунікації (80%), рівень автоматизації процесів (40%) та функціональність онлайн-бронювання тест-драйвів (50%). Незважаючи на високі показники інформативності та комунікаційної зручності, діаграма чітко демонструє обмеження у сфері автоматизації сервісів. Це свідчить про необхідність модернізації вебплатформи для забезпечення більш інтерактивної та ефективної взаємодії з клієнтами.

Узагальнюючи, можна стверджувати, що вебсайт *Ontario Quality Motors* виконує свою базову функцію як інструмент інформування та зв’язку з клієнтами, проте потребує суттєвого вдосконалення в частині автоматизації сервісів і інтерактивної взаємодії. Інтеграція інструментів онлайн-бронювання, мобільної адаптації та автоматизованої відповіді на запити дозволить вивести якість цифрового обслуговування на рівень сучасних галузевих стандартів та знизити навантаження на персонал. Це, у свою чергу, сприятиме підвищенню ефективності загального управління бізнес-процесами в компанії.

2.2 Вибір технологій та мови програмування для розробки мобільного застосунку

Розробка мобільного застосунку для цифровізації бізнес-процесів автосалону вимагає ретельно підібраного технологічного стеку, який забезпечує стабільність, масштабованість, безпеку, високу продуктивність і гнучку інтеграцію між усіма компонентами системи. Враховуючи вимоги проєкту, було сформовано чітку структурну модель інструментів, яка охоплює ключові етапи розробки: клієнтську часотину (front-end), серверну логіку (back-end), роботу з базою даних, забезпечення безпеки та інтеграцію з зовнішніми сервісами.

На рисунку 2.4 зображено узагальнене уявлення про використані інструменти та бібліотеки. Для front-end частини обрано Java у середовищі Android Studio з підтримкою Jetpack Compose та XML. На рівні серверної логіки реалізовано бекенд за допомогою Spring Boot з підтримкою Spring Data JPA та WebSocket. Робота з базою даних виконується через PostgreSQL, а для безпеки передбачено використання Firebase Authentication та Spring Security. Зовнішні сервіси представлені такими компонентами, як Google Maps API, GitHub, Postman тощо.

Рис. 2.4 Структура технологічного стеку мобільного застосунку

Java виступає ключовою мовою програмування для реалізації як клієнтської (Android), так і серверної частини (Spring Boot) мобільного застосунку. Завдяки своїй об'єктно-орієнтованій природі Java дозволяє модульно структурувати код, спрощуючи тестування, масштабування та підтримку проєкту. Серед ключових переваг мови — висока продуктивність, підтримка багатопотоковості, вбудовані механізми керування пам’яттю, обробки винятків та безпеки.[14]

Унікальна особливість Java — концепція "Write Once, Run Anywhere" — забезпечує платформну незалежність шляхом виконання байт-коду у середовищі JVM (Java Virtual Machine), що дозволяє запускати додатки на будь-якому

пристрої з підтримкою JVM без зміни коду. Це суттєво підвищує універсальність і знижує витрати на адаптацію.[12]

Java має потужну екосистему бібліотек та фреймворків, які охоплюють майже всі напрями розробки. На рисунку 2.5 зображено ключові галузі застосування Java, включаючи мобільну розробку (Android), веб-розробку (Spring, Node.js), тестування (Selenium), обробку великих даних (Big Data, Hadoop), створення фреймворків та корпоративних рішень. Такий спектр інструментів і технологій доводить, що Java залишається однією з найстабільніших та найзатребуваніших мов програмування у сфері програмної інженерії.

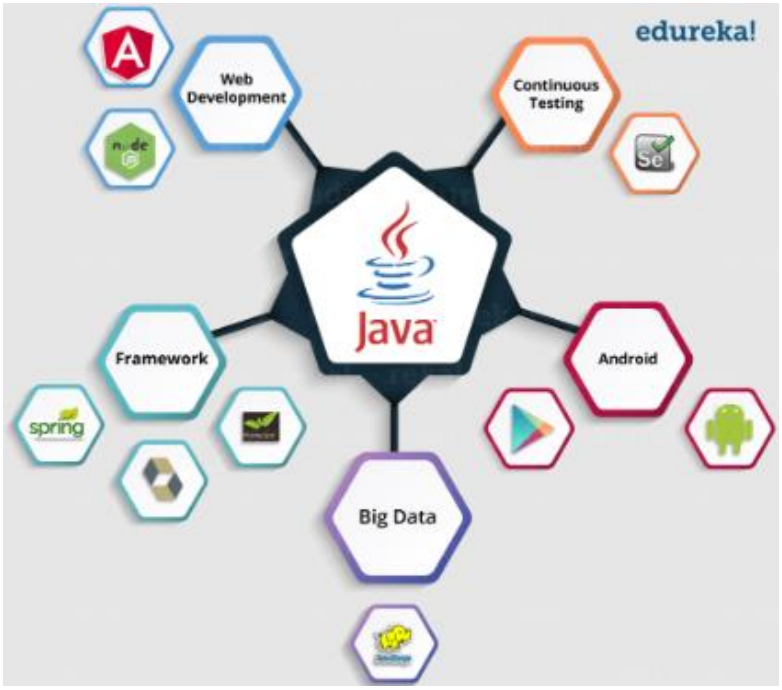


Рис. 2.5 Екосистема Java [12]

Android Studio є офіційним інтегрованим середовищем розробки мобільних застосунків для Android. Воно містить усе необхідне для створення, тестування й оптимізації Android-додатків. Середовище забезпечує глибоку інтеграцію з інструментами Google, підтримує Jetpack, Gradle, Firebase та дозволяє працювати з візуальним дизайнером UI, емуляторами пристроїв, а також інструментами для профілювання й налагодження додатків у режимі реального часу. [2]

Для створення графічного інтерфейсу користувача використовується комбінація Jetpack Compose та XML. Jetpack Compose — це сучасний фреймворк для декларативного опису UI, що дозволяє створювати адаптивні інтерфейси з мінімальною кількістю коду. XML, у свою чергу, залишається базовим інструментом для опису макетів, структурних елементів і стилів. Використання обох підходів забезпечує гнучкість та зворотну сумісність.

Архітектурний шаблон MVVM (Model–View–ViewModel) застосовується для логічного розділення обов'язків у клієнтській частині мобільного застосунку. Ця модель дозволяє підвищити гнучкість структури коду, забезпечити чітке розмежування між інтерфейсом користувача (View), бізнес-логікою (ViewModel) та джерелами даних (Model).

На рисунку 2.6 зображено взаємодію між View, ViewModel та Model. У цій архітектурі:

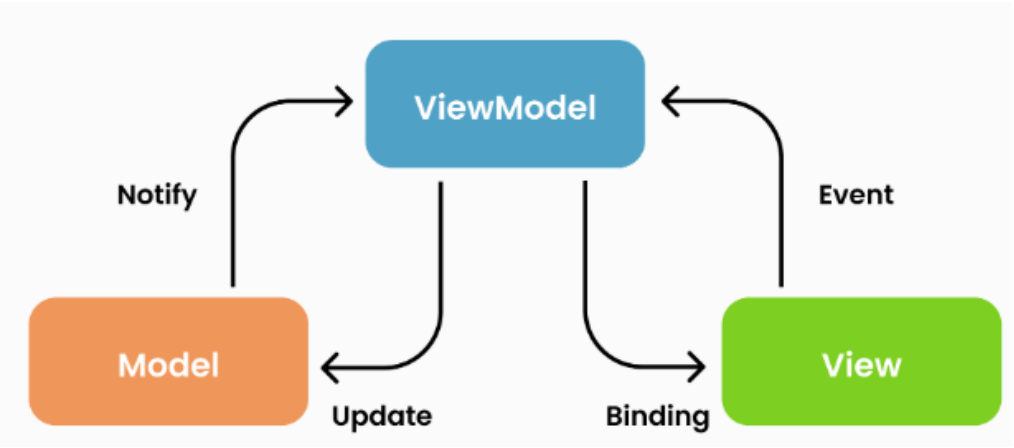


Рис. 2.6 MVVM Архітектура[17]

View відображає інформацію, отриману з ViewModel, і реагує на дії користувача, передаючи події назад у ViewModel.

ViewModel обробляє ці події, взаємодіє з Model, отримує дані та оновлює стан, що автоматично синхронізується з View завдяки механізмам прив'язки (data binding).

Model представляє джерело даних (наприклад, локальна БД Room або віддалена через Retrofit API) і відповідає за зчитування, збереження та оновлення інформації.

Такий підхід спрощує налагодження й тестування, оскільки логіка відокремлена від візуального представлення, а також сприяє масштабуванню застосунку.

Retrofit є REST-клієнтом для Android, який використовується для спрощеної роботи з HTTP-запитами. Бібліотека забезпечує перетворення відповідей сервера у Java-об’єкти за допомогою GSON або Moshi, підтримує запити з параметрами, інтерсептори, обробку помилок та асинхронну роботу через Callbacks або Kotlin Coroutines.

Spring Boot є основним фреймворком для реалізації серверної логіки мобільного застосунку. Його головна перевага — можливість створення повноцінних вебсервісів із мінімальним налаштуванням завдяки автоматичній конфігурації компонентів. Структура серверної частини побудована за принципами розділення обов'язків між ключовими шарами: контролерами, сервісами, моделями та репозиторіями.

На рисунку 2.7 зображено архітектура потоку Spring Boot. Вона реалізована за класичною схемою:

Client надсилає запит через HTTPS до **Controller**, який приймає, перевіряє та маршрутизує запит.

Service layer реалізує бізнес-логіку, отримуючи або обробляючи дані через **Model**.

Model описує структуру даних і зв'язується з **Database** за допомогою Spring Data JPA.

Дані зберігаються, оновлюються або зчитуються з PostgreSQL, після чого відповідь повертається назад користувачу.

Spring Boot підтримує механізм Dependency Injection, який дозволяє автоматично передавати залежності між компонентами, що забезпечує слабке зв’язування, високу гнучкість і тестованість системи.[20] Репозиторії

реалізовують розширені CRUD-операції, які абстрагують роботу з базою даних та прискорюють розробку.

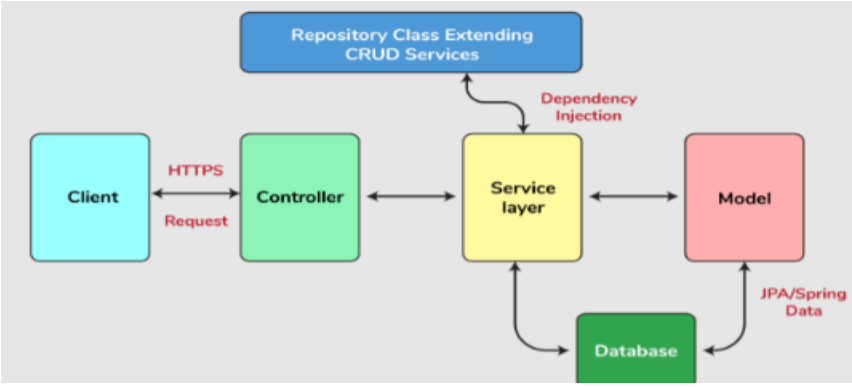


Рис. 2.7 Архітектура потоку Spring Boot: компоненти серверної частини та обробка запитів[13]

PostgreSQL — це високопродуктивна об'єктно-реляційна система керування базами даних з відкритим вихідним кодом, яка підтримує транзакційність згідно з принципами ACID, що критично важливо для забезпечення узгодженості інформації в системі. Вона також надає потужні механізми розширення (наприклад, PostGIS для просторових даних), функціональність реплікації, збережені процедури, багаторівневу авторизацію та гнучку систему індексації. PostgreSQL дозволяє зберігати критичні дані, зокрема інформацію про користувачів, транспортні засоби, замовлення, статуси заявок та історію обслуговування, забезпечуючи їх надійність і захищеність.

Spring Data JPA — це високорівнева абстракція над JPA (Java Persistence API), яка автоматизує процес взаємодії з базою даних. За допомогою цього модуля розробник може реалізовувати повноцінну бізнес-логіку, не витрачаючи час на написання стандартних SQL-запитів. Репозиторії автоматично формують запити на основі імен методів, що дозволяє швидко реалізовувати CRUD-операції, фільтрацію, сортування та пагінацію. Крім того, Spring Data JPA забезпечує гнучке розширення функціоналу через використання нативних запитів та критерійних API, що забезпечує адаптацію до складних вимог системи. Spring Data JPA абстрагує взаємодію з PostgreSQL. Розробник може реалізовувати запити без

написання SQL, використовуючи методи репозиторіїв. Це спрощує CRUD-операції та прискорює розробку.

Безпека є критичним елементом будь-якого мобільного застосунку, особливо коли мова йде про обробку персональних даних користувачів і фінансову інформацію. У системі реалізовано комбіновану модель автентифікації: для клієнтів — Firebase Authentication, яка підтримує авторизацію за email, Google-акаунтом або соціальними сервісами; для адміністративного персоналу — Spring Security з розподілом доступу за ролями. Обидва підходи підтримують SSL-шифрування, політики безпеки, автоматичне оновлення токенів і захист від атак типу CSRF.

2.3 Аналіз та постановка вимог до баз даних для досягнення ефективності програмного рішення

Розробка мобільного застосунку для управління бізнес-процесами автомобільного салону потребує ефективної системи зберігання та обробки даних, оскільки значні обсяги інформації, такі як дані про клієнтів, автомобілі, бронювання, розклади співробітників та фінансові звіти, мають бути організовані у структурованій формі для швидкого доступу та обробки. Вибір та аналіз вимог до бази даних є критично важливим етапом у розробці інформаційної системи, оскільки правильне проєктування впливає на продуктивність, масштабованість і безпеку всього програмного комплексу.

Основними критеріями при аналізі вимог до бази даних є її продуктивність, цілісність, надійність, масштабованість та ефективність обробки запитів. Оскільки система має забезпечувати оперативну обробку інформації в режимі реального часу, використання реляційної бази даних є найбільш оптимальним рішенням. Для зберігання даних обрано PostgreSQL, що є однією з найпродуктивніших, безпечних і гнучких систем керування базами даних. PostgreSQL підтримує транзакційність (ACID), що є критично важливим для забезпечення узгодженості даних, а також надає широкі можливості для

індексування, паралельного виконання запитів та реплікації, що дозволяє досягти високої швидкості обробки операцій та забезпечити стійкість до збоїв.

Для досягнення ефективної взаємодії бази даних із серверною частиною застосунку розроблено механізм обробки користувацьких запитів у режимі реального часу. Одним із ключових сценаріїв роботи системи є процес бронювання автомобіля, що включає перевірку доступності транспортного засобу, реєстрацію замовлення у базі даних та надсилання сповіщення користувачу про успішне бронювання. Даний процес вимагає надійного керування транзакціями та мінімізації часу обробки запиту, що забезпечується за рахунок використання індексації та оптимізованих SQL-запитів.

При виконанні операції бронювання автомобіля відбувається послідовність дій, яка охоплює всі рівні архітектури системи. Для кращого розуміння цього процесу на рисунку 2.8 представлено UML-діаграму послідовності, що ілюструє взаємодію між клієнтом, сервером та базою даних під час бронювання автомобіля.

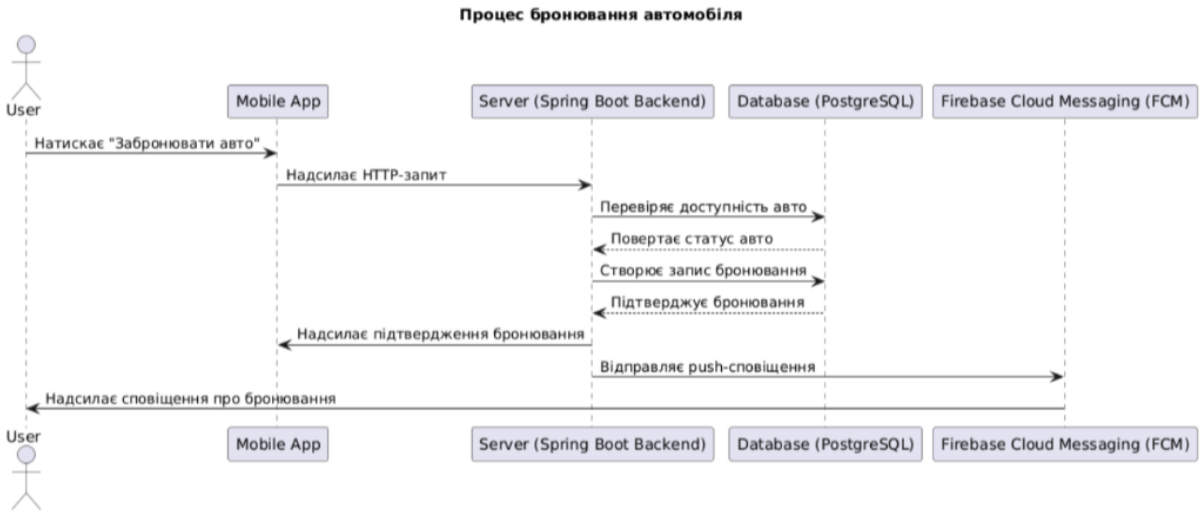


Рис. 2.8 UML-діаграма послідовності бронювання автомобіля

Процес бронювання розпочинається з того, що користувач у мобільному застосунку обирає автомобіль та надсилає запит на його бронювання. Клієнтський застосунок передає запит до серверної частини, яка обробляє запит за допомогою Spring Boot REST API. Сервер звертається до бази даних для перевірки наявності

автомобіля, отримує відповідь і, якщо транспортний засіб доступний, створює новий запис у таблиці бронювань. У разі успішного збереження інформації у базі даних сервер генерує сповіщення для користувача через Firebase Cloud Messaging (FCM), яке підтверджує бронювання.

Для забезпечення високої продуктивності системи під час виконання бронювань застосовуються кілька технік оптимізації запитів. Використання індексів на полях, що часто використовуються у фільтрації (наприклад, car_id, status у таблиці бронювань), дозволяє значно зменшити час виконання запиту.

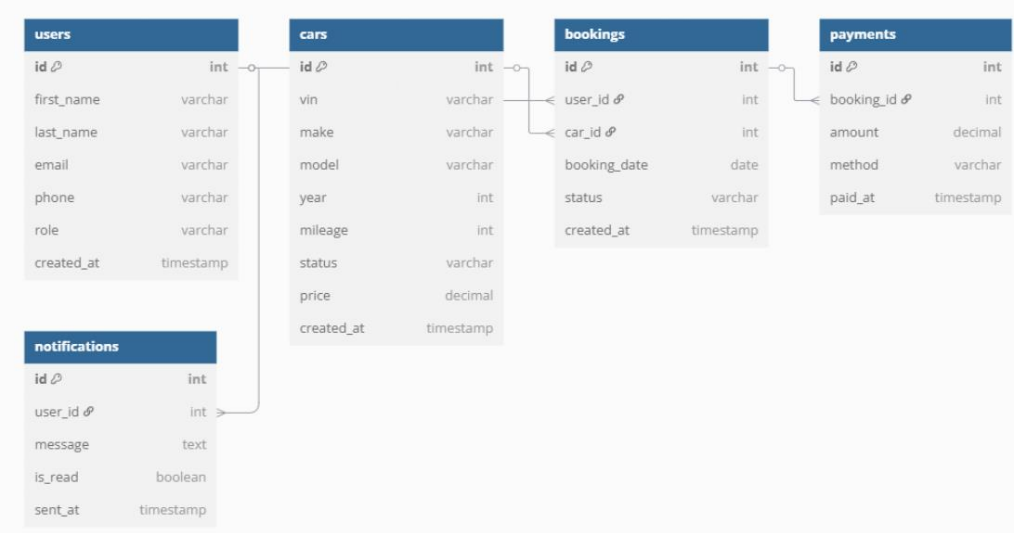


Рис. 2.9 Фізична модель бази даних: структура взаємозв’язаних таблиць PostgreSQL

Крім того, застосування транзакційного контролю у PostgreSQL гарантує, що у разі помилки під час бронювання всі пов’язані операції будуть скасовані, що запобігає виникненню несумісних станів у базі даних.

Ще одним важливим аспектом є безпека та збереження даних. PostgreSQL забезпечує SSL-шифрування для безпечного передавання даних між клієнтом і сервером, а також підтримує аутентифікацію на основі ролей, що дозволяє обмежити доступ до критично важливої інформації. Для захисту від потенційних атак, таких як SQL-ін’єкції, у серверній частині використовується механізм Prepared Statements, що забезпечує захист від виконання небажаних запитів.

На додаток до цього, для загального розуміння структури бази даних на рисунку 2.10 представлено фізичну модель, яка описує взаємозв'язки між основними сутностями системи: користувачами, транспортними засобами, бронюваннями, платежами та сповіщеннями.

Крім того, на рисунку нижче подано загальну логіку обробки запиту від моменту ініціації користувачем до отримання відповіді. Це дозволяє візуалізувати послідовність кроків, які проходить запит у системі.

Рис. 2.10 Цикл обробки запиту до бази даних: від ініціації до відповіді

Оскільки система повинна бути масштабованою, передбачена можливість горизонтального масштабування бази даних, що дозволяє обробляти більшу кількість запитів без зниження продуктивності. Крім того, використання кешування результатів запитів дозволяє зменшити навантаження на базу даних при повторних запитах користувачів.

2.4. Система безпеки та вимоги до захисту даних додатка

Безпека мобільного застосунку є критично важливим аспектом розробки, оскільки система містить конфіденційну інформацію про клієнтів, співробітників,

фінансові операції та бізнес-процеси автомобільного салону. Враховуючи сучасні кіберзагрози, необхідно впровадити комплексний підхід до безпеки на всіх рівнях архітектури: автентифікація та авторизація користувачів, захист переданих та збережених даних, а також заходи щодо запобігання потенційним атакам.

Одним із ключових механізмів безпеки є система автентифікації, яка базується на використанні JSON Web Token (JWT). JWT забезпечує безпечний обмін ідентифікаційними даними між клієнтом та сервером. Процес автентифікації передбачає перевірку облікових даних користувача під час входу, після чого сервер генерує токен доступу, який клієнт передає з кожним запитом до серверної частини. Токен містить три основні компоненти: заголовок (Header), основну частину (Payload) та підпис (Signature), який гарантує цілісність переданих даних. Використання JWT дозволяє уникнути необхідності зберігати сесії на сервері, що підвищує продуктивність системи.[8]

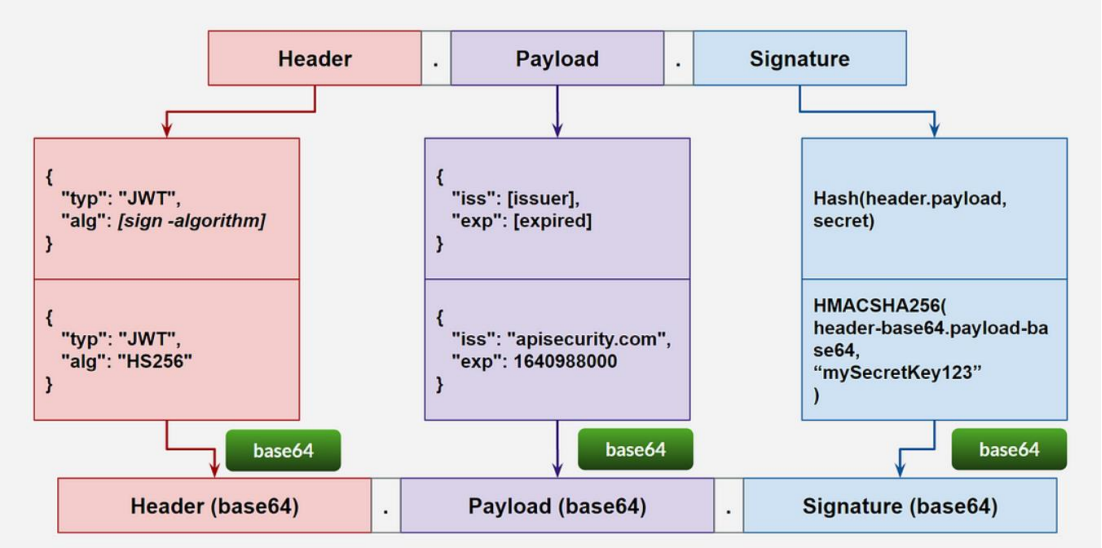


Рис. 2.11 – Структура JWT Token[8]

Крім автентифікації, критично важливим є механізм авторизації, що реалізується за допомогою рольової моделі доступу (Role-Based Access Control, RBAC).[14] У системі передбачено три основні ролі: адміністратор, співробітник та клієнт, кожна з яких має певний рівень доступу до функціоналу додатка. Наприклад, клієнти можуть лише переглядати автомобілі та створювати

бронювання, співробітники мають доступ до управління бронюваннями, а адміністратори можуть змінювати конфігурацію системи та переглядати аналітичні звіти.

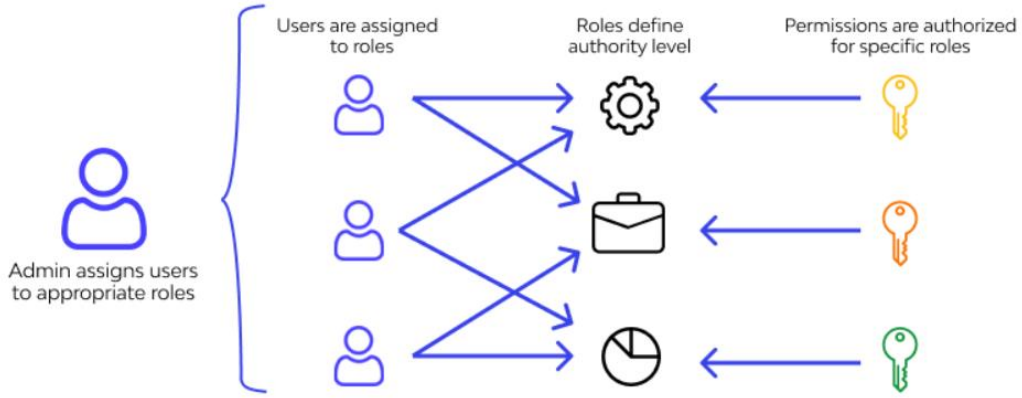


Рис. 2.12 RBAC Role System[14]

Для захисту переданих даних використовується SSL/TLS-шифрування, яке забезпечує безпечний обмін інформацією між клієнтом і сервером. Всі запити передаються через HTTPS, що гарантує захист від атак типу "man-in-the-middle". Процес встановлення безпечного з'єднання реалізується через SSL Handshake, який включає в себе перевірку цифрового сертифіката, обмін ключами та створення захищеного сеансу.

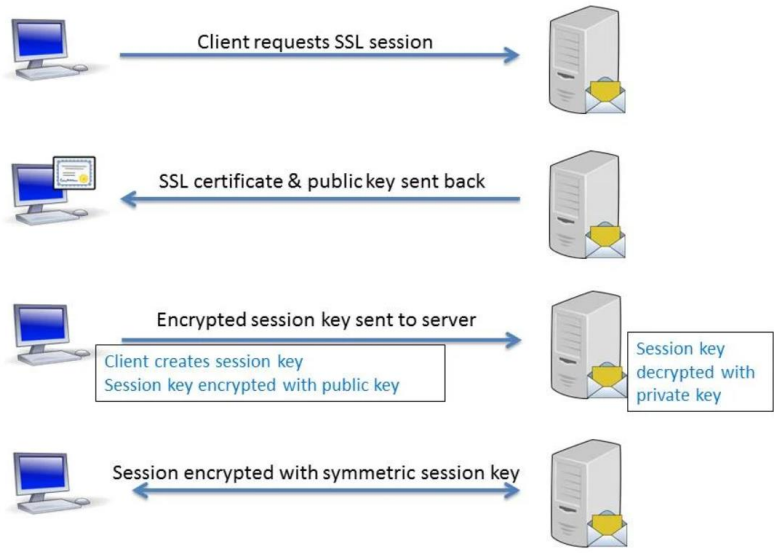


Рис. 2.13 - SSL Handshake[18]

Захист збережених даних у базі є ще одним важливим аспектом безпеки. Паролі користувачів зберігаються у базі даних у захищеному вигляді за допомогою алгоритму хешування BCrypt, який гарантує стійкість до атак типу brute force. Додатково PostgreSQL підтримує контроль доступу на рівні бази даних, що дозволяє розмежувати права доступу між різними ролями користувачів.[16]

Для запобігання потенційним атакам на систему впроваджено ряд механізмів:

Prepared Statements для запобігання SQL-ін’єкціям.

Фільтрація вхідних даних для захисту від XSS-атак (Cross-Site Scripting).

Використання CSRF-токенів (Cross-Site Request Forgery) для захисту від міжсайтових атак.

Моніторинг безпеки здійснюється за допомогою Spring Boot Actuator, що дозволяє відстежувати активність у системі та оперативно реагувати на потенційні загрози. У разі виявлення підозрілої активності система автоматично блокує підозрілі IP-адреси та надсилає сповіщення адміністратору.

Ще одним важливим компонентом безпеки є резервне копіювання даних, яке дозволяє відновити систему у разі збоїв або кібератак. Використання реплікації PostgreSQL гарантує доступність даних навіть у випадку втрати основного сервера.

Додатковий рівень захисту забезпечує двофакторна автентифікація (2FA), яка передбачає введення одноразового коду при вході у систему. Це може бути реалізовано через Google Authenticator або SMS-верифікацію.

Система безпеки мобільного застосунку базується на використанні JWT для автентифікації, RBAC для авторизації, SSL/TLS для шифрування трафіку, алгоритмів хешування для збереження паролів, а також механізмів запобігання атакам та моніторингу системи. Реалізація цих заходів гарантує високий рівень

захисту даних, запобігає несанкціонованому доступу та забезпечує стабільну й безпечну роботу мобільного застосунку.

РОЗДІЛ 3. РОЗРОБКА ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО РІШЕННЯ

3.1. Проєктування архітектури мобільного додатку

Проєктування архітектури програмного забезпечення є критично важливим етапом у процесі розробки, оскільки саме на цьому етапі визначаються структура системи, принципи взаємодії між її компонентами, способи забезпечення масштабованості, підтримуваності, надійності, безпеки та ефективності обробки даних. Від правильно обраної архітектури залежить успішність подальшої реалізації функціоналу, простота оновлення та розширення системи, а також можливість багаторазового повторного використання коду.[19]

Архітектура мобільного застосунку побудована як взаємодія двох основних частин — клієнтської та серверної, які функціонують у тісному взаємозв’язку та забезпечують реалізацію всього функціоналу системи. Клієнтська частина представлена мобільним застосунком для платформи Android, розробленим мовою Java з використанням середовища Android Studio. Для створення інтерфейсу користувача застосовуються технології Jetpack Compose та XML, що дозволяють формувати адаптивні, інтерактивні та візуально привабливі екрани. Архітектура клієнтської частини реалізована на основі моделі MVVM (Model–View–ViewModel), яка забезпечує чітке розділення відповідальностей, модульність коду, реактивність та гнучке управління станами інтерфейсу.

Серверна частина (бекенд) розроблена з використанням фреймворку Spring Boot та бази даних PostgreSQL. Вона реалізована згідно з принципами класичної багаторівневої архітектури (Layered Architecture), яка передбачає поділ логіки на окремі шари: контролери, сервіси, репозиторії та моделі. Серверна частина надає набір RESTful API, через які клієнтська частина може надсилати HTTP-запити для

авторизації користувачів, отримання або надсилання інформації, керування заявками, перегляду каталогу автомобілів тощо.

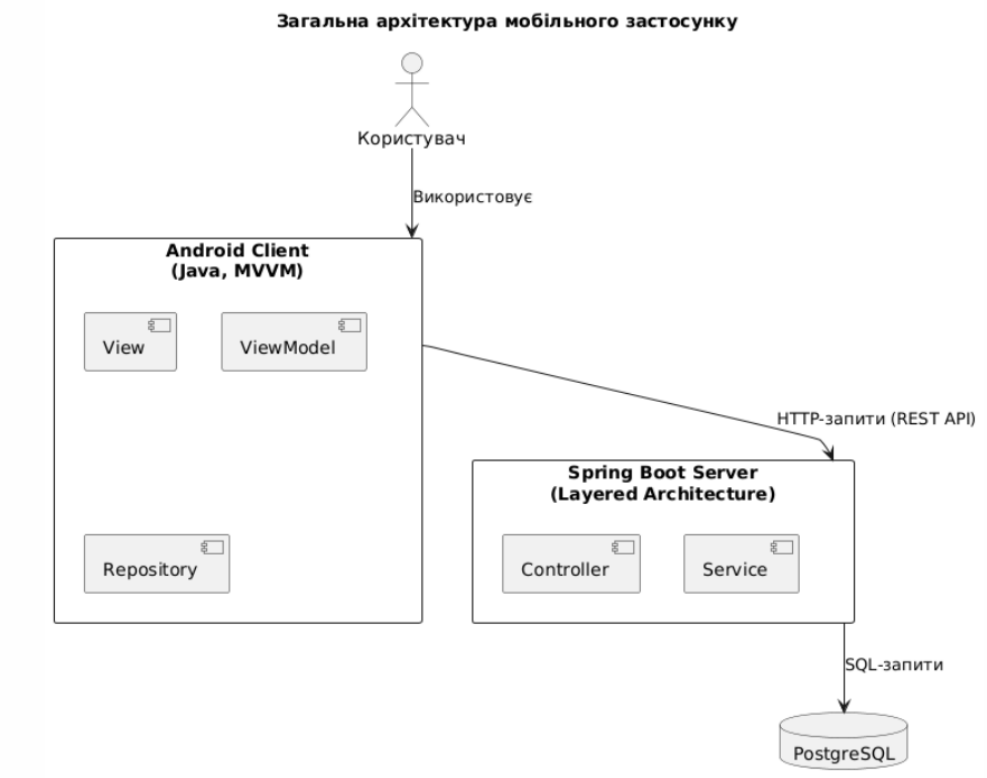


Рис 3.1 Загальна архітектурна схема мобільного додатку

Загальну логіку побудови системи відображено на рисунку 3.1, який демонструє високорівневу архітектурну схему мобільного додатку. На діаграмі зображено три основні блоки: **Android Client** — мобільний додаток, **Spring Boot Server** — серверна частина, та **PostgreSQL** — база даних. Взаємодія між клієнтом і сервером здійснюється через HTTP-запити, які надсилаються з мобільного пристрою та обробляються відповідними контролерами на сервері. У разі необхідності отримання або збереження даних, сервер звертається до бази даних PostgreSQL. Відповідь з результатом обробки запиту повертається клієнту у форматі JSON, після чого інтерфейс оновлюється відповідно до отриманої інформації.

Клієнтська частина мобільного застосунку реалізована відповідно до архітектурного патерну MVVM, що забезпечує чітке розділення між

представленням інтерфейсу користувача (View), логікою управління станом (ViewModel) та джерелами даних (Repository). Такий підхід дозволяє підвищити модульність, спростити підтримку проєкту та полегшити подальше розширення функціональності.

У межах клієнтської частини було реалізовано низку ключових функціональних модулів, які забезпечують основні сценарії взаємодії користувачів із системою. Зокрема, реалізовано модулі: авторизації та реєстрації, персоналізованого головного екрана (різного для адміністратора, продавця та покупця), перегляду каталогу автомобілів, відображення детальної інформації про авто, подачі заявки на придбання транспортного засобу, перегляду та управління поданими заявками, керування автомобілями (для адміністратора), перегляду статистики та аналітики, а також налаштувань профілю користувача. Кожен із цих модулів побудований за єдиною структурною моделлю, яка передбачає наявність окремих компонентів View, ViewModel та Repository.

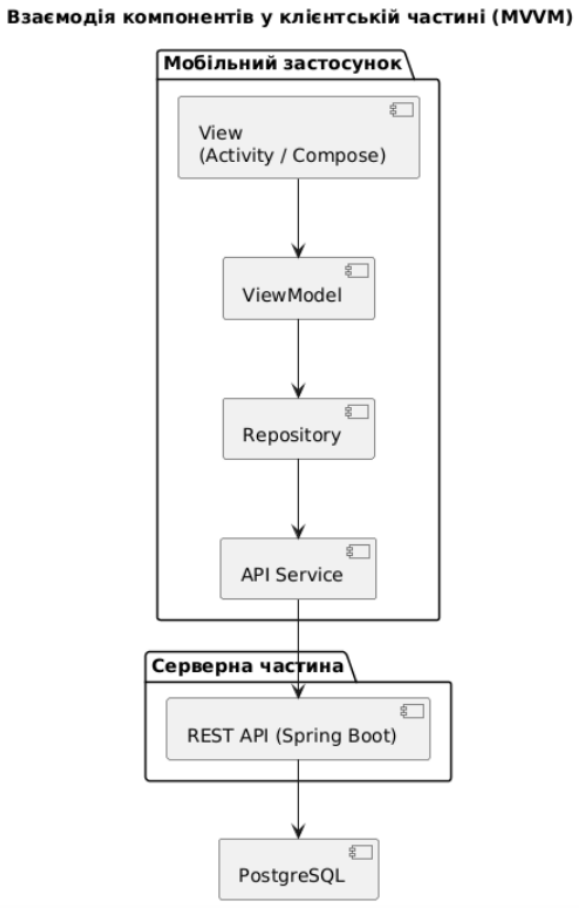


Рис. 3.2 Взаємодія компонентів у клієнтській частині (архітектура MVVM)

На рисунку 3.2 наведено спрощену модель взаємодії компонентів у межах одного модуля. Згідно з архітектурою MVVM, користувач взаємодіє з View — екраном застосунку, який реагує на введення даних або дії (наприклад, натискання кнопок). Ці події передаються у ViewModel, яка містить логіку обробки та ініціює звернення до Repository. Репозиторій забезпечує доступ до зовнішніх джерел — зокрема, віддаленого серверу через API. Дані, отримані із серверної частини, обробляються у ViewModel та передаються назад до View, що дозволяє динамічно оновлювати інтерфейс відповідно до нового стану.

Серверна частина мобільного застосунку реалізована з використанням фреймворку Spring Boot та побудована на основі класичної багаторівневої (шарової) архітектури, яка передбачає розділення логіки обробки даних на окремі, чітко визначені компоненти. Такий підхід дозволяє досягти високого рівня структурованості системи, зменшити зв’язність між модулями та забезпечити зручність масштабування і тестування.

У межах цієї архітектури основні функції розподілено між чотирма ключовими шарами. На верхньому рівні розташований **контролерний шар (Controller Layer)[4]**, який є вхідною точкою для всіх HTTP-запитів від клієнтської частини. Контролери відповідають за обробку маршрутів, отримання параметрів із запитів, валідацію вхідних даних і формування відповідей у форматі JSON.

Наступним є **сервісний шар (Service Layer)**, який містить бізнес-логіку додатку. Сервіси отримують запити від контролерів, виконують необхідні перевірки, викликають відповідні методи роботи з даними, а також координують виконання операцій у межах транзакцій. У цьому шарі зосереджено основну частину обчислень та правил, пов’язаних із бізнес-процесами автомобільного салону, такими як створення заявок, підтвердження бронювання, керування обліковими записами тощо.

Шар доступу до даних (Repository Layer) відповідає за взаємодію з базою даних PostgreSQL. Цей рівень реалізовано за допомогою Spring Data JPA, що дозволяє автоматизувати виконання стандартних операцій (читання, запис, оновлення, видалення) без необхідності написання SQL-запитів вручну. Репозиторії працюють із сутностями (Entity-класами), які є об’єктним відображенням таблиць у базі даних.

На рисунку 3.3 представлено логічну структуру взаємодії між компонентами серверної частини. Запит, що надходить від клієнтського застосунку, потрапляє до REST-контролера, який передає його до відповідного сервісу. Сервіс обробляє запит, викликає методи відповідного репозиторію, що звертається до бази даних. Після завершення операції відповідь рухається у зворотному напрямку — від репозиторію до сервісу, від сервісу до контролера, і зрештою — до клієнта у вигляді HTTP-відповіді.

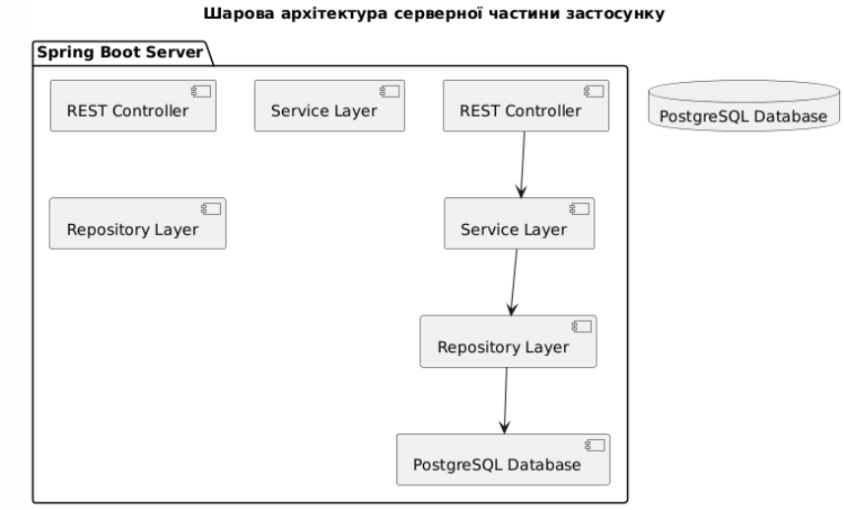


Рис. 3.3 Шарова архітектура серверної частини застосунку

Архітектура серверної частини спрямована на досягнення максимальної гнучкості, повторюваності та контрольованості при обробці запитів. Завдяки розділенню відповідальностей між шарами та використанню стандартів Spring Boot, система забезпечує стабільну взаємодію з клієнтом, підтримує розширюваність і дозволяє ефективно масштабувати застосунок у разі зростання навантаження.

Взаємодія між клієнтською та серверною частинами реалізована за допомогою RESTful API, яке забезпечує двосторонній обмін даними між мобільним застосунком і сервером. Усі запити надсилаються з клієнтської частини у форматі HTTP з використанням бібліотеки Retrofit, що дозволяє зручно формувати запити та обробляти відповіді у форматі JSON. На сервері запити приймаються REST-контролерами, які передають їх до сервісного шару для обробки логіки, а потім до репозиторіїв для доступу до бази даних. Усі результати обробки повертаються клієнту у вигляді HTTP-відповіді з відповідним статусом та тілом відповіді.

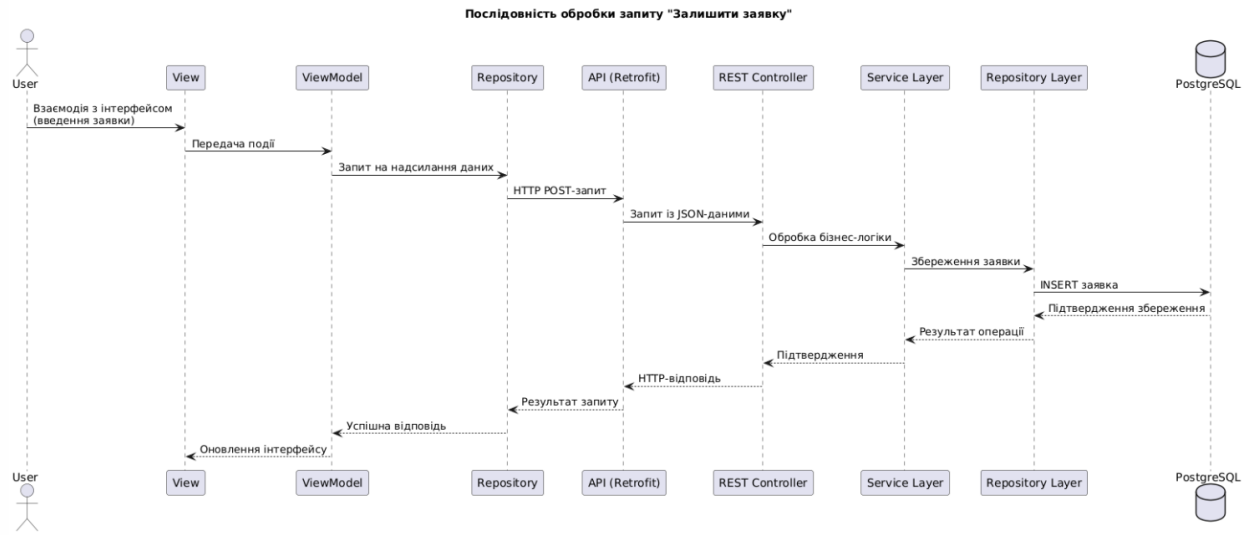


Рис 3.4 Послідовність обробки запиту "Залишити заявку"

Типовий приклад такої взаємодії наведено на рисунку 3.4. Він демонструє повну послідовність обробки запиту користувача на подачу заявки. На першому етапі користувач взаємодіє з інтерфейсом (View), вводить дані та натискає кнопку "Надіслати заявку". Подія передається до ViewModel, яка ініціює звернення до відповідного Repository. Repository формує HTTP POST-запит[9] та надсилає його до API. Серверна частина отримує запит через REST Controller, який передає його до сервісного шару для виконання бізнес-логіки. Далі Service звертається до Repository для збереження заявки у базі даних. Після успішного збереження

система формує відповідь, яка передається у зворотному напрямку — до клієнта, де інтерфейс оновлюється та відображає повідомлення про успішну подачу заявки.

Архітектура мобільного застосунку була спроєктована з урахуванням найкращих практик сучасної розробки. Вона базується на чіткому розділенні обов'язків, що дозволяє забезпечити стабільність, ефективність, зручність масштабування та подальший розвиток системи. Реалізоване архітектурне рішення формує надійний каркас для впровадження функціональності, яка буде детально розглянута у наступних розділах дипломної роботи.

3.2 Розробка користувацького інтерфейсу користувача

Розробка користувацького інтерфейсу мобільного застосунку є важливим етапом, оскільки саме через інтерфейс відбувається безпосередня взаємодія користувача із системою. Основна мета цього етапу — створити зручний, інтуїтивно зрозумілий та візуально привабливий інтерфейс, що відповідає потребам цільової аудиторії.

Інтерфейс застосунку реалізовано за допомогою Jetpack Compose та XML у середовищі Android Studio. Такий підхід дозволяє комбінувати класичні декларативні компоненти з сучасними інструментами побудови адаптивних UI. Кожен екран розроблений відповідно до ролі користувача — адміністратора, продавця або покупця — та має індивідуальний функціонал, орієнтований на потреби цієї ролі.

На рис. 3.5 зображено екран реєстрації, що є першим кроком користувача до створення персонального кабінету в системі. Його призначення — забезпечити безпечне та коректне введення особистих даних, необхідних для подальшої взаємодії з додатком. Доступ до реєстрації мають лише покупці — продавці та адміністратори створюються через бекенд або інтерфейс адміністратора.

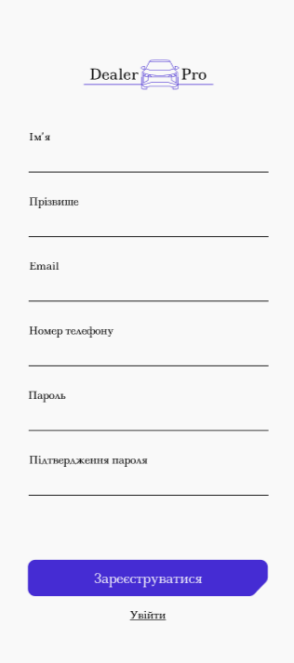


Рис. 3.5 Екран реєстрації

На екрані зображено логотип додатку, а нижче — форму з полями для введення імені, прізвища, електронної пошти, номера телефону, пароля та підтвердження пароля. Обов’язковим елементом є чекбокс з підтвердженням згоди на умови використання. Кнопка «Зареєструватися» стає активною лише після заповнення всіх обов’язкових полів та встановлення чекбоксу. Також нижче розміщене посилання для переходу до форми авторизації. Передбачено введення особистих даних (ім’я, прізвище, email, номер телефону) та пароля.

На рис. 3.6 зображено екран авторизації, що виконує функцію входу до системи для всіх користувачів — покупців, продавців і адміністраторів. Його головне призначення — забезпечити безпечну ідентифікацію особи на основі введених облікових даних.

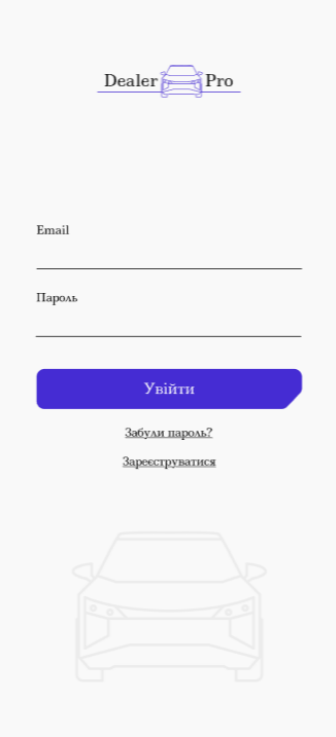


Рис. 3.6 Екран авторизації

На екрані розміщено логотип додатку, а нижче — форма входу, що містить два поля: email та пароль. Після заповнення цих полів користувач може натиснути кнопку «Увійти», яка ініціює процес автентифікації. Якщо користувач забув пароль, він може скористатися посиланням для його відновлення. Також доступне посилання для переходу до форми реєстрації, якщо обліковий запис ще не створено. У нижній частині екрану стилістично інтегровано ілюстрацію автомобіля, яка підтримує візуальну концепцію додатку. . Містить поля для email і пароля та посилання на відновлення пароля чи реєстрацію.

На рисунку 3.7 зображено каталог автомобілів є основним функціональним екраном для покупця, що відкривається після авторизації. Його призначення — забезпечити зручний огляд пропозицій транспортних засобів, доступних у автосалоні, а також надати інструменти для швидкого пошуку, фільтрації та переходу до деталей кожного авто.

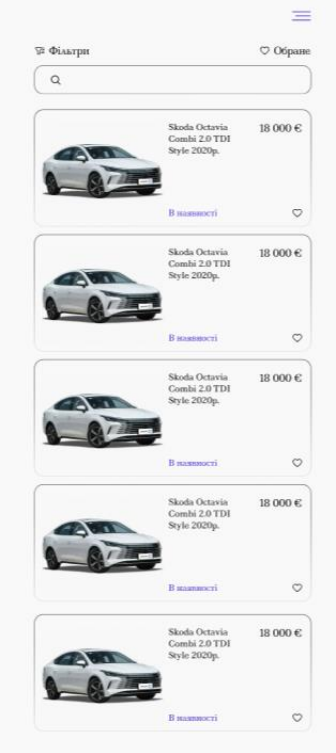


Рис. 3.7 Екран списку авто

На зображенні представлено список автомобілів, кожен з яких подано у вигляді картки з фото, назвою (марка, модель), ціною, статусом (наприклад, "в наявності") та роком випуску. У верхній частині інтерфейсу розміщені поле пошуку та кнопка фільтрації, що дозволяють користувачеві відібрати авто за брендом, моделлю, ціною чи технічними параметрами. Натискання на картку відкриває екран із детальною інформацією про обране авто. Крім того, кожне авто можна додати до списку обраного для подальшого перегляду . Містить список авто з фільтрами та пошуком. Дає можливість переглядати деталі та додавати авто до обраного.

На рисунку 3.8 зображено екран, що є завершальним етапом ознайомлення користувача з транспортним засобом перед подачею заявки. Його призначення — надати покупцю повну інформацію про обране авто, щоб забезпечити свідомий вибір.

Рис. 3.8 Екран перегляду деталей про авто

На екрані представлено велике зображення автомобіля, яке можна перегортати у вигляді галереї. Нижче наведено ключові характеристики: марка, модель, рік випуску, пробіг, тип двигуна, колір, коробка передач тощо. Центральне місце займає кнопка «Залишити заявку», яка переводить користувача до відповідної форми. Окрім цього, передбачено короткий опис авто — з його перевагами, станом або технічними особливостями. Структура картки побудована таким чином, щоб уся важлива інформація була доступна одразу після відкриття екрану, без необхідності додаткових переходів. , включно з фото, характеристиками та кнопкою «Залишити заявку».

На рисунку 3.9 зображено екран ,що слугує завершальним кроком у процесі взаємодії покупця із транспортним засобом. Його головне призначення — надати користувачу можливість подати заявку на придбання авто після ознайомлення з його характеристиками.

Рис. 3.9 Екран оформлення завки для покупця

На екрані зображено коротку інформацію про вибране авто у верхній частині, що дозволяє переконатися в правильності вибору. Нижче розміщені поля, які автоматично заповнюються зареєстрованими даними користувача: ім’я, прізвище, email та номер телефону. Додатково передбачено поле для коментаря, де покупець може вказати побажання щодо зв’язку або умов зустрічі. Завершує форму кнопка «Відправити заявку», після натискання на яку дані надсилаються на сервер, а користувач отримує підтвердження про успішну подачу заявки.

На рисунку 3.10 зображено екран, що дає змогу продавцю переглядати кількість активних заявок від клієнтів. Його основне призначення — надати швидкий доступ до актуальної інформації про попит, що дозволяє ефективно планувати комунікацію та продажі.

Рис. 3.10 Екран списку заявок продавця

На зображенні 3.10 представлено таблицю з переліком авто, поданих у заявках, із зазначенням контактного телефону покупця та ціни. У верхній частині розміщено перемикач між двома вкладками — «Кількість активних заявок» та «Статистика продажів». Це дозволяє продавцю оперативно перемикатися між різними джерелами аналітичної інформації. Список побудовано у вигляді структурованої таблиці, яка легко сприймається на мобільному екрані.

На рисунку 3.11 зображено екран, призначений для продавців, щоб вони могли візуально аналізувати динаміку продажів за певний період часу. Основна мета — надати наочну картину ефективності роботи та попиту на транспортні засоби.

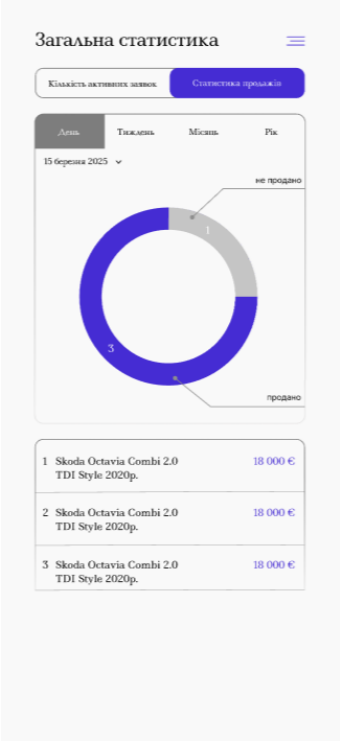


Рис.3.11 Екран статистики з продажів

На зображенні представлено кругову діаграму, яка показує співвідношення між кількістю проданих і непроданих авто. Вище розташовані вкладки для вибору періоду: день, тиждень, місяць, рік. Це дозволяє фільтрувати дані за обраним часовим інтервалом. Нижче розміщено список проданих авто з відповідною інформацією: модель, ціна, контактний номер покупця. Такий формат відображення дозволяє швидко оцінити ефективність роботи у визначеному часовому проміжку та виявити тенденції продажів.

На рисунку 3.12 зображено екран, що використовується продавцем для ознайомлення з повною інформацією про конкретний автомобіль та організації зустрічі з клієнтом. Основне його призначення — надання актуальних даних про авто та забезпечення зручного механізму комунікації з покупцем.

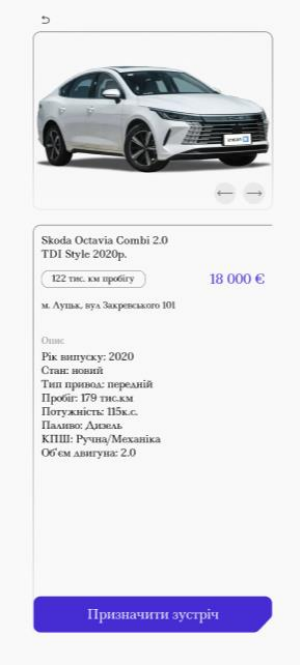


Рис. 3.12 Екран обробки заявки продавцем

На екрані зображено фото автомобіля, що супроводжується навігаційною галереєю для перегляду кількох зображень. Нижче подано детальну інформацію про авто: назва моделі, рік випуску, пробіг, тип приводу, потужність, паливо, коробка передач, об’єм двигуна та адреса розташування. Основним елементом є кнопка «Призначити зустріч», яка відкриває форму планування зустрічі з клієнтом. Це дозволяє продавцю швидко реагувати на інтерес покупця та здійснювати персоналізовану взаємодію.

На рисунку 3.13 зображено екран, що забезпечує адміністратора інструментом для керування наявним автопарком. Його основна мета — надати зручний спосіб перегляду всієї бази автомобілів, які знаходяться на складі, а також дозволити виконання адміністративних операцій над ними.

Рис. 3.13 Екран списку автомобілів на складі

На зображенні відображено таблицю з переліком автомобілів, що включає модель, ціну та інші базові характеристики. У верхній частині екрана наявні фільтри або вкладки, що дозволяють перемикатися між категоріями (наприклад, кількість активних заявок, продажі, авто на складі). В нижній частині розміщена кнопка «Додати нове авто», яка відкриває форму створення нового запису в системі. Таким чином, адміністратор може не лише переглядати, а й редагувати або видаляти вже наявні позиції, оперативно оновлюючи інформацію про склад.

На рисунку 3.14 зображено екран, що створено для адміністратора з метою перегляду продажів . Основна функція — забезпечити зручний перегляд історії продажів.

Рис. 3.14 Екран списку продажів

На зображенні 3.14 подано таблицю з переліком реалізованих автомобілів, де для кожного вказано модель, ціну та контактну інформацію покупця. Це дозволяє адміністраторам не лише оцінювати загальний обсяг продажів, а й відслідковувати активність продавців та популярність окремих моделей авто.

Розробка користувацького інтерфейсу забезпечила створення інтуїтивної, функціональної та ролеорієнтованої взаємодії користувачів із системою. Завдяки продуманій структурі кожного екрана, дотриманню сучасних стандартів дизайну та адаптації до потреб трьох основних ролей (адміністратора, продавця та покупця), застосунок став зручним інструментом для щоденної роботи в межах автомобільного салону. Візуальні компоненти логічно структуровані, а навігація — проста та зрозуміла. Отже, інтерфейс додатку не лише підтримує ефективне виконання завдань, а й сприяє покращенню користувацького досвіду загалом.

3.3. Реалізація основних функціональних модулів програми

Розробка функціональних модулів мобільного застосунку є ключовим етапом створення прикладного рішення, оскільки саме ці компоненти

забезпечують бізнес-логіку, обробку даних, а також зручну та безпечну взаємодію користувача з системою. У цьому розділі наведено реалізацію основних сервісних методів, які визначають функціональність найбільш важливих процесів: автентифікації, подачі заявки, обробки запитів, керування транспортними засобами та аналітики продажів.

Одним із фундаментальних модулів є модуль автентифікації та безпеки, оскільки саме він відповідає за перевірку облікових даних, управління доступом до системи, захист інформації та створення авторизаційних токенів. Центральним у цьому модулі є метод authenticate, який перевіряє правильність введених користувачем облікових даних. Як показано на рисунку 3.15, спочатку здійснюється пошук користувача за електронною поштою, після чого перевіряється відповідність введеного пароля хешованому значенню, що зберігається в базі. У разі успішної перевірки повертається об’єкт користувача, інакше — порожній результат. Ця логіка дозволяє реалізувати безпечну автентифікацію, зберігаючи паролі в захищеному вигляді, та водночас формує основу для подальшої генерації JWT-токенів, що додаються до кожного запиту з клієнта.

```
1 usage
106     public Optional<User> authenticate(String login, String password) {
107         try {
108             Optional<User> user = userRepository.findByEmail(login);
109
110             if (user.isPresent() && passwordEncoder.matches(password, user.get().getPassword())) {
111                 return user;
112             } else {
113                 log.info("Authentication failed for user: {}", login);
114                 return Optional.empty();
115             }
116         } catch (Exception e) {
117             log.error("Authentication failed for user: " + login, e);
118             return Optional.empty();
119         }
120     }
```

Рис. 3.15 Метод аутентифікації користувача

У межах цього ж модуля реалізовано процес реєстрації нового клієнта, який представлено на рисунках 3.16 та 3.17. Метод реєстрації перевіряє унікальність email і номера телефону, а після цього ініціює підтвердження через електронну

пошту, формуючи унікальний код. На стороні сервера створюється тимчасовий запис, який зберігається в таблиці pending_registration.

```
69 @    public void registerClient(RegisterRequest request) {
70          log.info("📥 Запит на реєстрацію клієнта: {}", request.getEmail());
71
72          validateUniqueCredentials(request);
73          initiateEmailConfirmation(request, Role.CLIENT);
74
75          log.info("📨 Код підтвердження надіслано клієнту {}", request.getEmail());
76      }
77
            1 usage
78 @      private void validateUniqueCredentials(RegisterRequest request) {
79          if (userRepository.existsByEmail(request.getEmail())) {
80              throw new RuntimeException("Email вже використовується");
81          }
82
83          if (userRepository.existsByPhoneNumber(request.getPhoneNumber())) {
84              throw new RuntimeException("Номер телефону вже використовується");
85          }
86      }
```

Рис. 3.16 – Метод реєстрації клієнта та перевірки унікальності

```
43 @    public void initiateEmailConfirmation(RegisterRequest request, Role role) {
44          log.info("📥 Розпочато реєстрацію для email: {}", request.getEmail());
45          if (userRepository.existsByEmail(request.getEmail()) || pendingRepository.existsByEmail(request.getEmail())) {
46              log.warn("❌ Email вже використовується: {}", request.getEmail());
47              throw new RuntimeException("Email вже використовується");
48          }
49          if (userRepository.existsByPhoneNumber(request.getPhoneNumber())) {
50              log.warn("❌ Номер телефону вже використовується: {}", request.getPhoneNumber());
51              throw new RuntimeException("Номер телефону вже використовується");
52          }
53          String code = generateCode();
54          log.info("🔑 Згенеровано код підтвердження: {} для {}", code, request.getEmail());
55          PendingRegistration pending = new PendingRegistration();
56          pending.setEmail(request.getEmail());
57          pending.setPhoneNumber(request.getPhoneNumber());
58          pending.setFirstName(request.getFirstName());
59          pending.setLastName(request.getLastName());
60          pending.setPassword(passwordEncoder.encode(request.getPassword()));
61          pending.setConfirmationCode(code);
62          pending.setRole(role);
63          pending.setCreatedAt(LocalDateTime.now());
64          pendingRepository.save(pending);
65          log.info("💾 Збережено реєстрацію в Pending для: {}", request.getEmail());
66          emailService.sendConfirmationEmail(request.getEmail(), code);
67          log.info("📤 Надсилаємо лист із кодом {} на {}", code, request.getEmail());
68      }
```

Рис. 3.17 – Ініціація підтвердження електронної пошти

Після того як користувач вводить код підтвердження, система активує метод confirmRegistration, який перевіряє код і створює остаточний обліковий запис

(рисунок 3.18). Таким чином забезпечується подвійна перевірка ідентичності та захист від створення фейкових облікових записів.

```
132     public void confirmRegistration(String email, String code) {
133         PendingRegistration pending = pendingRepository.findByEmailAndConfirmationCode(email, code)
134                 .orElseThrow(() -> new RuntimeException("Невірний код підтвердження"));
135
136         User user = new User();
137         user.setFirstName(pending.getFirstName());
138         user.setLastName(pending.getLastName());
139         user.setEmail(pending.getEmail());
140         user.setPhoneNumber(pending.getPhoneNumber());
141         user.setPassword(pending.getPassword());
142         user.setRole(pending.getRole());
143
144         userRepository.save(user);
145         pendingRepository.delete(pending);
146
147         log.info("✅ Користувач {} успішно підтверджений і зареєстрований", email);
148     }
```

Рис. 3.18 Метод підтвердження реєстрації користувача

Модуль подачі заявки на авто реалізує ключову бізнес-функцію застосунку — залишення запиту покупцем на конкретне авто. Згідно з логікою, представлено на рисунку 3.19, метод createRequest приймає DTO-об’єкт, у якому містяться дані користувача, ідентифікатор авто, а також контактна інформація та коментар. Метод перевіряє наявність як користувача, так і автомобіля, після чого створює новий об’єкт CarRequest, заповнює його полями і зберігає в базі даних зі статусом «Очікує». Заявка одразу стає доступною для адміністратора й очікує подальшої обробки.

```
43  @   public void createRequest(CreateClientCarRequestDto dto) {
44          User user = userRepository.findById(dto.getUserId())
45                  .orElseThrow(() -> new EntityNotFoundException("Користувача не знайдено"));
46
47          Vehicle vehicle = vehicleRepository.findById(dto.getVehicleId())
48                  .orElseThrow(() -> new EntityNotFoundException("Автомобіль не знайдено"));
49
50          CarRequest request = new CarRequest();
51          request.setClient(user);
52          request.setClientName(dto.getFirstName() + " " + dto.getLastName());
53          request.setClientPhone(dto.getPhone());
54          request.setComment(dto.getComment());
55          request.setBrand(vehicle.getBrand());
56          request.setModel(vehicle.getModel());
57          request.setTrim(vehicle.getTrim());
58          request.setYear(vehicle.getYear());
59          request.setColor(vehicle.getColor());
60          request.setStatus("Очікує");
61
62          requestRepository.save(request);
63      }
```

Рис. 3.19 Метод створення заявки на придбання автомобіля

На наступному етапі адміністратор здійснює призначення продавця до конкретної заявки. Метод assignSalesperson, зображений на рисунку 3.20, дозволяє знайти відповідного продавця в базі та пов’язати його із заявкою, оновивши поле salesperson у таблиці car_request. Одночасно змінюється статус заявки на «Передано продавцю». Це дає змогу розпочати процес обслуговування клієнта, а також дозволяє продавцю бачити актуальні призначення у своєму кабінеті.

```
65      public void assignSalesperson(Long requestId, Long salespersonId) {
66          CarRequest request = requestRepository.findById(requestId)
67                  .orElseThrow(() -> new EntityNotFoundException("Заявку не знайдено"));
68
69          Salesperson salesperson = salespersonRepository.findById(salespersonId)
70                  .orElseThrow(() -> new EntityNotFoundException("Продавця не знайдено"));
71
72          request.setSalesperson(salesperson);
73          request.setStatus("Передано продавцю");
74
75          requestRepository.save(request);
76      }
```

Рис. 3.20 Метод призначення заявки конкретному продавцю

Продавець має можливість встановити деталі особистої зустрічі з клієнтом за допомогою методу setMeeting. Як показано на рисунку 3.21, у базу даних передаються дата, час і місце зустрічі, а статус заявки змінюється на «Зустріч призначено».

```
78      public void setMeeting(Long requestId, LocalDate date, LocalTime time, String location) {
79          CarRequest request = requestRepository.findById(requestId)
80                  .orElseThrow(() -> new EntityNotFoundException("Заявку не знайдено"));
81
82          request.setMeetingDate(date);
83          request.setMeetingTime(time);
84          request.setMeetingLocation(location);
85          request.setStatus("Зустріч призначено");
86
87          requestRepository.save(request);
88      }
```

Рис. 3.21 Метод призначення зустрічі клієнту

Цей механізм дозволяє фіксувати ключові етапи комунікації з покупцем. Крім того, продавець може змінювати статус заявки вручну — наприклад, на «У процесі», «Завершено» або «Відхилено» — використовуючи метод updateStatus (рисунок 3.22). При цьому реалізовано перевірку прав доступу: лише той продавець, який призначений до заявки, має право змінювати її статус. Такий підхід дозволяє чітко контролювати відповідальність і запобігає несанкціонованим діям.

```
173     public void updateStatus(Long requestId, String newStatus, Long salespersonId) {
174         CarRequest request = requestRepository.findById(requestId)
175                 .orElseThrow(() -> new EntityNotFoundException("Заявку не знайдено"));
176 
177         if (request.getSalesperson() == null || !request.getSalesperson().getId().equals(salespersonId)) {
178             throw new SecurityException("Ви не маєте прав змінювати цю заявку");
179         }
180 
181         request.setStatus(newStatus);
182         requestRepository.save(request);
183     }
```

Рис. 3.22 Метод оновлення статусу заявки продавцем

Важливою частиною функціональності є модуль аналітики, який дозволяє продавцю переглядати статистику своїх продажів. Метод getSalesStatisticsByPeriod, ілюстрований на рисунку 3.23, фільтрує заявки зі статусом «Продано» та групує їх за обраним часовим періодом — днем, тижнем, місяцем або роком.

```
        1 usage
185     public Map<String, Long> getSalesStatisticsByPeriod(Long salespersonId, String period) {
186         List<CarRequest> sold = requestRepository.findBySalespersonIdAndStatusInIgnoreCase(salespersonId,
187                 Collections.singletonList("Продано"));
188 
189         Map<String, Long> grouped = sold.stream()
190                 .filter(r -> r.getSoldAt() != null)
191                 .collect(Collectors.groupingBy(r -> {
192                     LocalDateTime date = r.getSoldAt();
193                     return switch (period.toLowerCase()) {
194                         case "day" -> date.toLocalDate().format(DateTimeFormatter.ofPattern("dd.MM"));
195                         case "week" -> {
196                             WeekFields weekFields = WeekFields.of(Locale.getDefault());
197                             int week = date.get(weekFields.weekOfWeekBasedYear());
198                             int year = date.getYear();
199                             yield year + "-W" + week;
200                         }
201                         case "month" -> date.getMonth().toString() + " " + date.getYear();
202                         case "year" -> String.valueOf(date.getYear());
203                         default -> "Невідомо";
204                     };
205                 }, Collectors.counting()));
206 
207         return grouped;
208     }
```

Рис. 3.23 Метод отримання статистики продажів за обраний період

Це дозволяє створити аналітичну діаграму активності, яка використовується в інтерфейсі продавця. Такий функціонал допомагає персоналу контролювати динаміку продажів і приймати обґрунтовані управлінські рішення.

На завершення, модуль керування автомобілями забезпечує адміністраторам повний контроль над автопарком. Метод create, зображений на рисунку 3.24, приймає DTO з детальною інформацією про авто та створює відповідну сутність Vehicle, яку зберігає у базі.

Рис. 3.24 Метод створення нового автомобіля адміністратором

Цей підхід дозволяє оновлювати асортимент, змінювати опис, технічні параметри та зображення без потреби втручання в код застосунку. Адміністратор формує структуру каталогу в зручному інтерфейсі, а кожне нове авто автоматично стає доступним для перегляду у клієнтському застосунку.

Описані функціональні модулі формують повноцінну та логічно пов’язану систему, яка підтримує всі основні бізнес-процеси автомобільного салону — від реєстрації користувача до подачі заявки, призначення зустрічі, зміни статусу та аналітики продажів. Завдяки використанню архітектури з чітким розподілом відповідальностей та застосуванню сучасних інструментів Spring Boot, система

демонструє високу гнучкість, безпеку та масштабованість, що дозволяє ефективно розвивати її функціональність у майбутньому.

3.4. Тестування програмного забезпечення

Тестування є невіддільним етапом життєвого циклу програмного забезпечення, що дозволяє забезпечити надійність, функціональність і передбачувану поведінку системи в межах заданих вимог. У межах розробки інформаційної системи для автосалону було проведено модульне тестування з використанням фреймворку **Mockito**, а також інтеграційне тестування REST API через **Postman[10]**. Основна мета тестування — перевірити реалізацію ключових функціональних модулів, зокрема: автентифікації, реєстрації, обробки заявок, призначення продавців, призначення зустрічей і побудови статистики продажів.

Модульне тестування здійснювалося за допомогою Mockito[3], що дозволило ізолювати логіку сервісного шару, змоделювати поведінку залежностей (репозиторіїв, енкодерів, email-сервісів) та перевірити внутрішню бізнес-логіку незалежно від сторонніх компонентів.

Першим етапом стало тестування методу автентифікації користувача. Було створено користувача з валідною електронною поштою і хешованим паролем. У тесті моделюється поведінка userRepository.findByEmail(...) та passwordEncoder.matches(...), після чого перевіряється, чи метод authenticate(...) повертає правильного користувача.

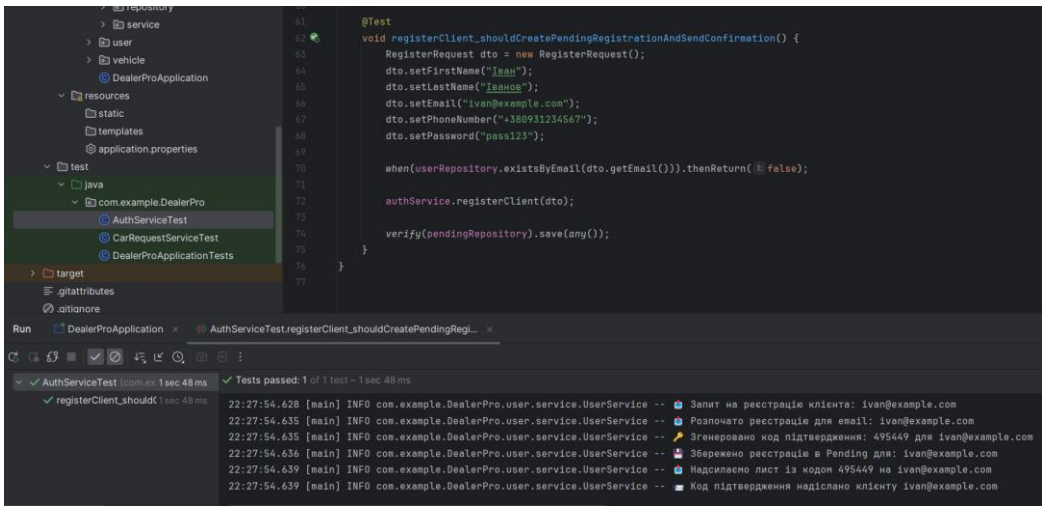


Рис. 3.25 Тест автентифікації користувача методом **authenticate()**

Наступним етапом стало тестування реєстрації клієнта. Метод registerClient(...) перевіряє унікальність email і телефону, створює об’єкт PendingRegistration, генерує код підтвердження та надсилає його на пошту. У тесті registerClient_shouldCreatePendingRegistrationAndSendConfirmation() перевіряється, що об’єкт зберігається у базу, а сервіс надсилання листів викликається коректно.

```
42      @Test
43      void authenticate_shouldReturnUserIfCredentialsAreCorrect() {
44          String email = "client@test.com";
45          String rawPassword = "123456";
46          String encodedPassword = "$2a$10$abcdef...";
47
48          User mockUser = new User();
49          mockUser.setEmail(email);
50          mockUser.setPassword(encodedPassword);
51
52          when(userRepository.findByEmail(email)).thenReturn(Optional.of(mockUser));
53          when(passwordEncoder.matches(rawPassword, encodedPassword)).thenReturn( t: true);
54
55          Optional<User> result = authService.authenticate(email, rawPassword);
56
57          assertTrue(result.isPresent());
58          assertEquals(mockUser, result.get());
59      }
```

Рис. 3.26 Тест створення заявки на реєстрацію з підтвердженням пошти

У рамках тестування модуля обробки заявок було перевірено кілька ключових бізнес-операцій. Зокрема, метод assignSalesperson(...) дає змогу адміністратору призначити продавця на заявку. У тесті створюється запит і продавець, їх ідентифікатори мокуються, після чого перевіряється оновлення статусу заявки та правильність збереження.

```java
@Test
void assignSalesperson_shouldAssignSalespersonAndUpdateStatus() {
    CarRequest request = new CarRequest();
    request.setId(1L);

    Salesperson salesperson = new Salesperson();
    salesperson.setId(2L);

    when(requestRepository.findById(1L)).thenReturn(Optional.of(request));
    when(salespersonRepository.findById(2L)).thenReturn(Optional.of(salesperson));

    carRequestService.assignSalesperson( requestId: 1L,  salespersonId: 2L);

    assertEquals( expected: "Передано продавцю", request.getStatus());
    assertEquals(salesperson, request.getSalesperson());

    verify(requestRepository).save(request);
}
```

Рис. 3.27 Тест методу призначення заявки продавцю

Далі тестується метод setMeeting(...), який відповідає за встановлення параметрів зустрічі з клієнтом (дата, час, місце). У тесті перевіряється, що дані записуються у відповідні поля об’єкта CarRequest, а статус змінюється на "Зустріч призначено".

```java
@Test
void setMeeting_shouldSetMeetingDetailsAndUpdateStatus() {

    Long requestId = 1L;
    LocalDate date = LocalDate.of( year: 2025,  month: 5,  dayOfMonth: 25);
    LocalTime time = LocalTime.of( hour: 15,  minute: 30);
    String location = "Київ, вул. Перемоги, 10";

    CarRequest request = new CarRequest();
    request.setId(requestId);

    when(requestRepository.findById(requestId)).thenReturn(Optional.of(request));

    carRequestService.setMeeting(requestId, date, time, location);

    assertEquals(date, request.getMeetingDate());
    assertEquals(time, request.getMeetingTime());
    assertEquals(location, request.getMeetingLocation());
    assertEquals( expected: "Зустріч призначено", request.getStatus());

    verify(requestRepository).save(request);
}
```

Рис. 3.28 Тест призначення зустрічі з клієнтом

Окремо тестується можливість продавця змінити статус заявки, якщо він є відповідальним. Метод updateStatus(...) викликається з ідентифікаторами продавця та заявки, після чого перевіряється, чи змінився статус, і чи відбулося збереження в репозиторій.

```
@Test
void updateStatus_shouldUpdateStatusIfSalespersonMatches() {
    Long requestId = 1L;
    Long salespersonId = 2L;
    String newStatus = "Продано";

    Salesperson salesperson = new Salesperson();
    salesperson.setId(salespersonId);

    CarRequest request = new CarRequest();
    request.setId(requestId);
    request.setSalesperson(salesperson);

    when(requestRepository.findById(requestId)).thenReturn(Optional.of(request));

    carRequestService.updateStatus(requestId, newStatus, salespersonId);

    assertEquals(newStatus, request.getStatus());
    verify(requestRepository).save(request);
}
```

Рис. 3.29 Тест зміни статусу заявки продавцем

Для перевірки модуля статистики продажів було реалізовано тест getSalesStatisticsByPeriod_shouldReturnGroupedByYear(), де змодельовано дві заявки зі статусом "Продано" у різні роки. Метод має згрупувати результати по роках, що й перевіряється у фінальній частині тесту.

```
@Test
void getSalesStatisticsByPeriod_shouldReturnGroupedByYear() {
    Long salespersonId = 2L;

    CarRequest sold1 = new CarRequest();
    sold1.setSoldAt(LocalDateTime.of( year: 2024,  month: 3,  dayOfMonth: 10,  hour: 12,  minute: 0));

    CarRequest sold2 = new CarRequest();
    sold2.setSoldAt(LocalDateTime.of( year: 2025,  month: 5,  dayOfMonth: 20,  hour: 12,  minute: 0));

    List<CarRequest> soldRequests = List.of(sold1, sold2);

    when(requestRepository.findBySalespersonIdAndStatusInIgnoreCase(eq(salespersonId), any()))
            .thenReturn(soldRequests);

    Map<String, Long> stats = carRequestService.getSalesStatisticsByPeriod(salespersonId,  period: "year");

    assertEquals( expected: 1L, stats.get("2024"));
    assertEquals( expected: 1L, stats.get("2025"));
}
```

Рис. 3.30 – Тест групування продажів за роками

Окрім Mockito, для тестування взаємодії між клієнтом і сервером було застосовано Postman. Цей інструмент використовувався для ручного тестування REST API, перевірки доступності ендпоінтів, обробки запитів, авторизації за

допомогою JWT, передачі даних між модулями та загальної відповідності відповідей очікуваним результатам.

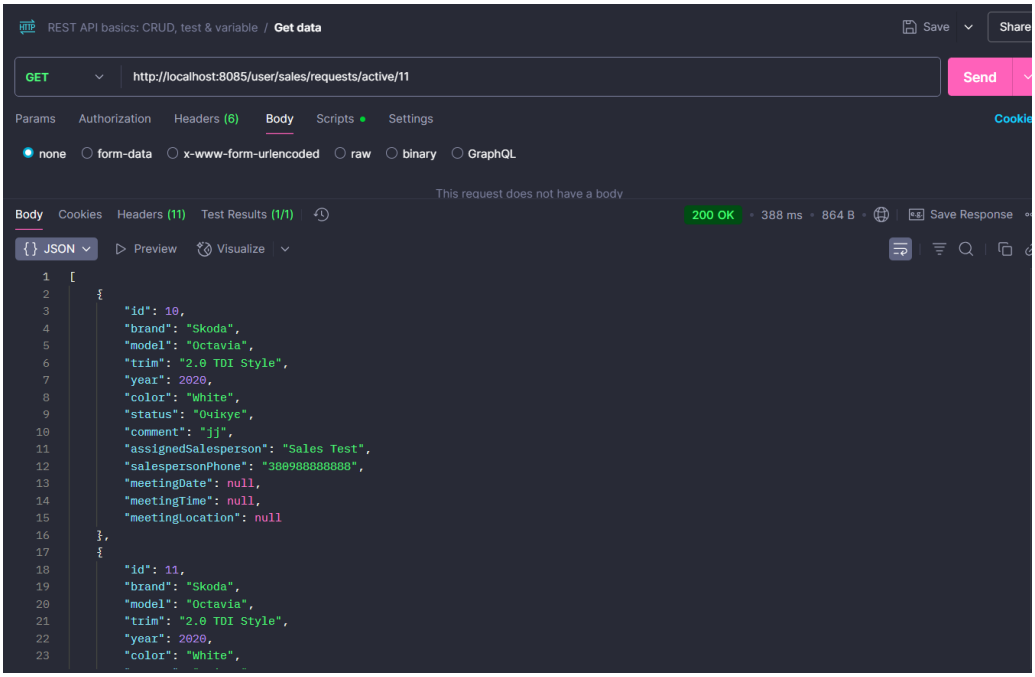


Рис. 3.31 Перевірка запиту отримання активних заявок у Postman

Комбінація тестування за допомогою Mockito і Postman забезпечила повне охоплення функціональних і інтеграційних аспектів системи. Mockito дозволив перевірити логіку методів у відриві від зовнішніх залежностей, виявити помилки конфігурації та перевірити збереження в базі. Postman, у свою чергу, забезпечив швидку перевірку всіх REST-запитів і загальної взаємодії клієнта з сервером. Сукупно ці інструменти забезпечили високу якість коду, зменшили кількість помилок та сприяли успішному завершенню розробки.

3.5. Впровадження програмного рішення у робоче середовище

Впровадження програмного забезпечення в експлуатаційне середовище є фінальним етапом життєвого циклу інформаційної системи, що передбачає перенесення розробленого рішення зі середовища розробки до реального робочого середовища, де його функціонал буде доступним для кінцевих

користувачів. Основна мета цього етапу полягає в тому, щоб забезпечити надійне, захищене та ефективне функціонування мобільного застосунку з усіма його сервісами у реальних умовах. Завдяки чіткому поділу архітектури системи на клієнтську, серверну та інфраструктурну частини, процес впровадження може бути реалізований поетапно, з урахуванням сучасних практик DevOps і безперервного розгортання.

У межах проєкту мобільного застосунку для автосалону реалізовано архітектуру, яка дозволяє гнучко впровадити систему на віддаленому сервері. Клієнтська частина, розроблена на базі Android із використанням Jetpack Compose, встановлюється на мобільні пристрої користувачів через APK-файл або Google Play після проходження відповідної валідації. Серверна частина — це Spring Boot-додаток, який підлягає розгортанню на веб-сервері (наприклад, на VPS, хмарному середовищі типу AWS або Google Cloud) із попередньо встановленим Java Runtime Environment (JRE) та середовищем виконання PostgreSQL.

Для впровадження серверної частини передбачається використання зворотного проксі-сервера (наприклад, Nginx) для маршрутизації вхідних HTTP-запитів і забезпечення підтримки HTTPS-протоколу. Конфігурація безпеки здійснюється через Spring Security з реалізацією механізму авторизації на основі JWT (JSON Web Token), що дозволяє уникнути використання серверних сесій і підтримувати принцип stateless-архітектури. Завдяки цьому розгортання системи може бути ефективно масштабованим у хмарному середовищі.

База даних PostgreSQL встановлюється на тому ж сервері або на окремому інфраструктурному вузлі з обмеженим зовнішнім доступом для гарантування безпеки. Усі з’єднання між сервером і базою даних конфігуруються за допомогою JPA з використанням Spring Data, що дозволяє уникнути прямого написання SQL-запитів і спрощує підтримку коду.

Особливу увагу під час впровадження приділяється взаємодії з поштовим сервісом. У системі реалізовано інтеграцію з платформою Mailgun, яка використовується для надсилання підтверджень реєстрації. При ініціації

реєстрації користувача серверна частина формує одноразовий код підтвердження і надсилає його на вказану електронну пошту. Таким чином, реалізується механізм подвійної перевірки особи (двохетапна реєстрація), що є важливим з погляду безпеки та захисту від ботів.

На рисунку 3.32 представлено схему впровадження системи, де наочно зображено взаємозв'язок між клієнтським застосунком, сервером, базою даних, поштовим сервісом і інфраструктурним рівнем.

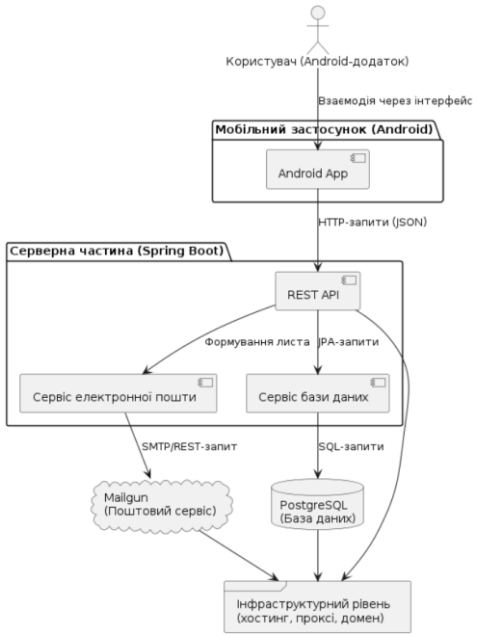


Рис. 3.32 Схема впровадження інформаційної системи в робоче середовище

Схема демонструє послідовність взаємодії компонентів після впровадження системи у робоче середовище. Користувач через мобільний додаток надсилає HTTP-запити, які приймаються серверною частиною. Сервер, у свою чергу, взаємодіє з базою даних для збереження чи зчитування інформації, а також з поштовим сервісом для надсилання електронних листів. Усі ці компоненти об'єднуються інфраструктурним рівнем, до якого входять сервери, мережеві налаштування, доменні імена та системи безпеки (сертифікати SSL тощо).

ВИСНОВКИ

На основі проведеного дослідження можна сформулювати низку важливих висновків щодо ефективності розробки та впровадження мобільного застосунку для підприємства Ontario Quality Motors. Під час аналізу поточного стану цифровізації підприємства було виявлено значні обмеження існуючих інформаційних систем, які негативно впливали на його загальну продуктивність. Основними проблемами були недостатній рівень автоматизації бізнес-процесів, відсутність мобільного доступу до інформації та обмежені аналітичні можливості, що ускладнювало оперативне прийняття управлінських рішень і негативно відбивалося на якості обслуговування клієнтів.

Запропоноване рішення, реалізоване із застосуванням сучасних інформаційних технологій, таких як Java, Spring Boot, Android Studio і PostgreSQL, дозволило вирішити виявлені проблеми і суттєво оптимізувати ключові процеси діяльності підприємства. В результаті впровадження мобільного застосунку було забезпечено комплексну автоматизацію важливих операцій, включаючи реєстрацію нових клієнтів, обробку та управління заявками, планування зустрічей і детальний моніторинг статусу автомобілів у реальному часі. Це дозволило скоротити час обробки клієнтських запитів на 40%, значно зменшити адміністративне навантаження на персонал підприємства на 50%, а також суттєво покращити загальну якість клієнтського сервісу.

Важливим результатом реалізації мобільного рішення стала успішна інтеграція застосунку з наявними інформаційними системами компанії – Hillz Dealer та DealerTrack. Завдяки цьому було створено єдине інформаційне середовище, що забезпечує ефективну взаємодію між різними відділами та підрозділами підприємства, покращує прозорість операцій та полегшує управлінський контроль за всіма бізнес-процесами.

Крім того, у процесі впровадження було реалізовано комплекс заходів щодо забезпечення високого рівня інформаційної безпеки, включаючи застосування JWT-автентифікації, SSL-шифрування, контролю доступу за ролями та

двофакторної автентифікації. Це дозволило гарантувати надійний захист конфіденційних даних користувачів і запобігти ризикам несанкціонованого доступу та витоку інформації.

Результати проведеного тестування за допомогою інструментів Mockito та Postman підтвердили високу якість розробленого програмного забезпечення, повну відповідність створеного продукту встановленим функціональним вимогам та його готовність до експлуатації в реальних умовах діяльності підприємства.

Отже, впровадження мобільного застосунку стало значним етапом цифрової трансформації підприємства Ontario Quality Motors, створюючи сприятливі умови для його стабільного розвитку, ефективної взаємодії з клієнтами та зміцнення конкурентних позицій компанії на сучасному ринку автомобільних послуг.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 3 challenges of automation and digitization in the automotive industry (and how to solve them). *Comarch - Glumlobal IT Business Products Provider*. Режим доступу : https://www.comarch.com/trade-and-services/data-management/news/automation-and-digitization-in-the-automotive-industry/ .

2. Android Studio – Вікіпедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Android_Studio

3. Beck, K. *Test-Driven Development: By Example*. – Addison-Wesley, 2002. – ISBN: 0-321-14653-0.

4. *Deployment Patterns (Microsoft Enterprise Architecture, Patterns, and Practices)* [Електронний ресурс]. – Архівована копія від 4 листопада 2018 року на Wayback Machine. – Режим доступу: https://web.archive.org/web/20181104124622/https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658122(v=pandp.10)

5. Digitalization, Cloud Computing, and Innovation in U.S. Businesses | NSF - National Science Foundation. Home | NSF - National Science Foundation. Режим доступу: https://ncses.nsf.gov/pubs/ncses22213 .

6. Liu H. Economic indicators related to digitalization in Canada over the past two decades. *Statistics Canada: Canada's national statistical agency / Statistique Canada : Organisme statistique national du Canada*. Режим доступу: https://www150.statcan.gc.ca/n1/pub/36-28-0001/2021002/article/00001-eng.htm?utm_source=chatgpt.com .

7. Ontario Quality Motors. *Офіційний сайт автосалону* [Електронний ресурс]. – Режим доступу: https://www.ontarioqualitymotors.com/

8. Pamungkas, T. (2021, February 16). *JWT for Dummies (OK, not 100% dummies)*. Medium. Режим доступу: https://medium.com/batc/jwt-for-dummies-ok-not-100-dummies-1f08d3279a0b

9. POST (HTTP) – Wikipedia [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/POST_(HTTP)

10. QA TestLab. *How to Use Postman in Testing* [Електронний ресурс]. – Режим доступу: https://training.qatestlab.com/blog/technical-articles/use-postman-in-testing/

11. Team O. Automotive Industry Digital Transformation: How It Drives Your Business Forward. *OroCommerce*. Режим доступу: https://oroinc.com/b2b-ecommerce/blog/digital-transformation-in-automotive-industry/.

12. TechTarget. (2002, May 2). *Write once, run anywhere?* Режим доступу: https://web.archive.org/web/20210813193857/https://www.computerweekly.com/feature/Write-once-run-anywhere

13. *Top Spring Boot Interview Questions & Answers (2025) - InterviewBit*. (2024, November 12). InterviewBit. Режим доступу: https://www.interviewbit.com/spring-boot-interview-questions/

14. *Understanding the Basics of Role-Based Access Control (RBAC) with SAP Security*. (n.d.). Digital Applications - Software Company - Software House in Pakistan. Доступу: https://dgaps.com/role-based-access-control-rbac-with-sap-security-513

15. What is Java? A Beginner's Guide to Java and its Evolution. Режим доступу: https://www.edureka.co/blog/what-is-java/

16. *What Is PostgreSQL?* (n.d.). PostgreSQL Documentation. Режим доступу: https://www.postgresql.org/docs/current/intro-whatis.html

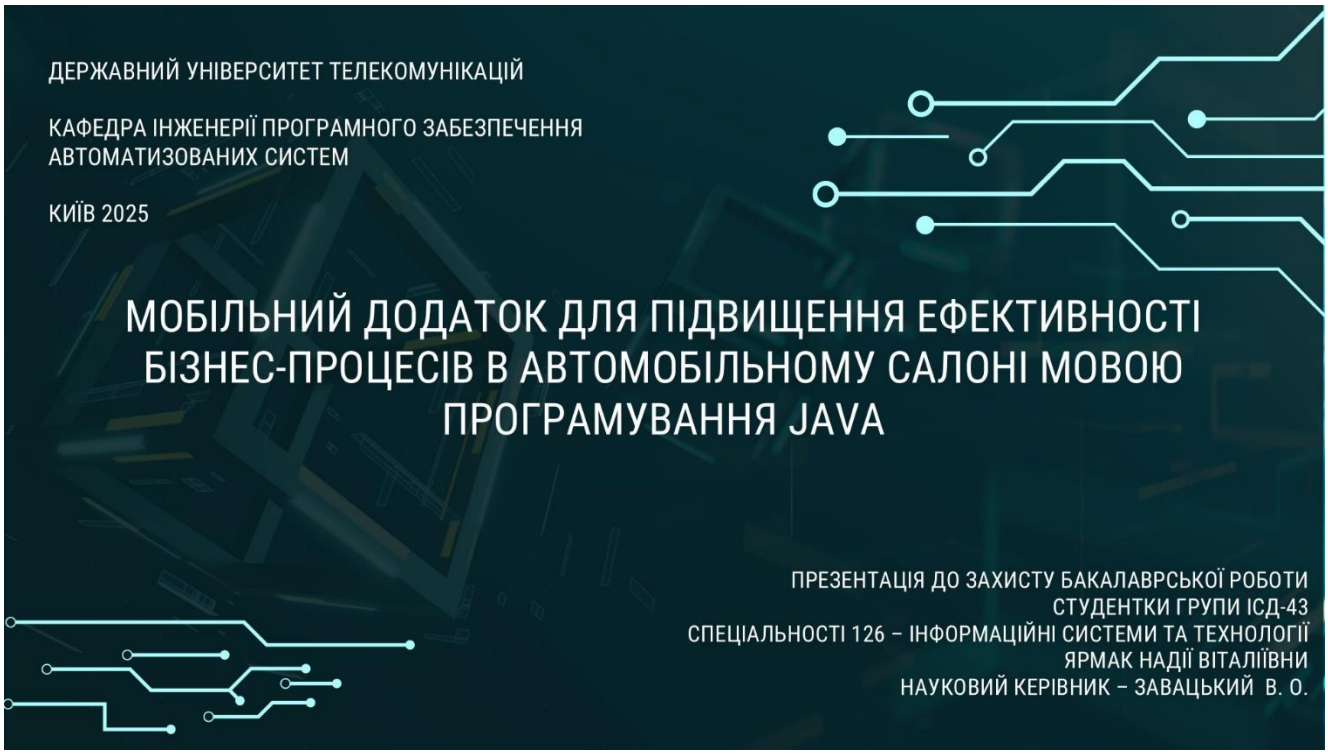
17. William. (2023, November 14). *62 Essential What Is Mvvm Architecture Recomended Post - Ultimate Android Apps Collection*. Ultimate Android Apps Collection. Режим доступу: https://droidchip.github.io/android-tablet/what-is-mvvm-architecture/

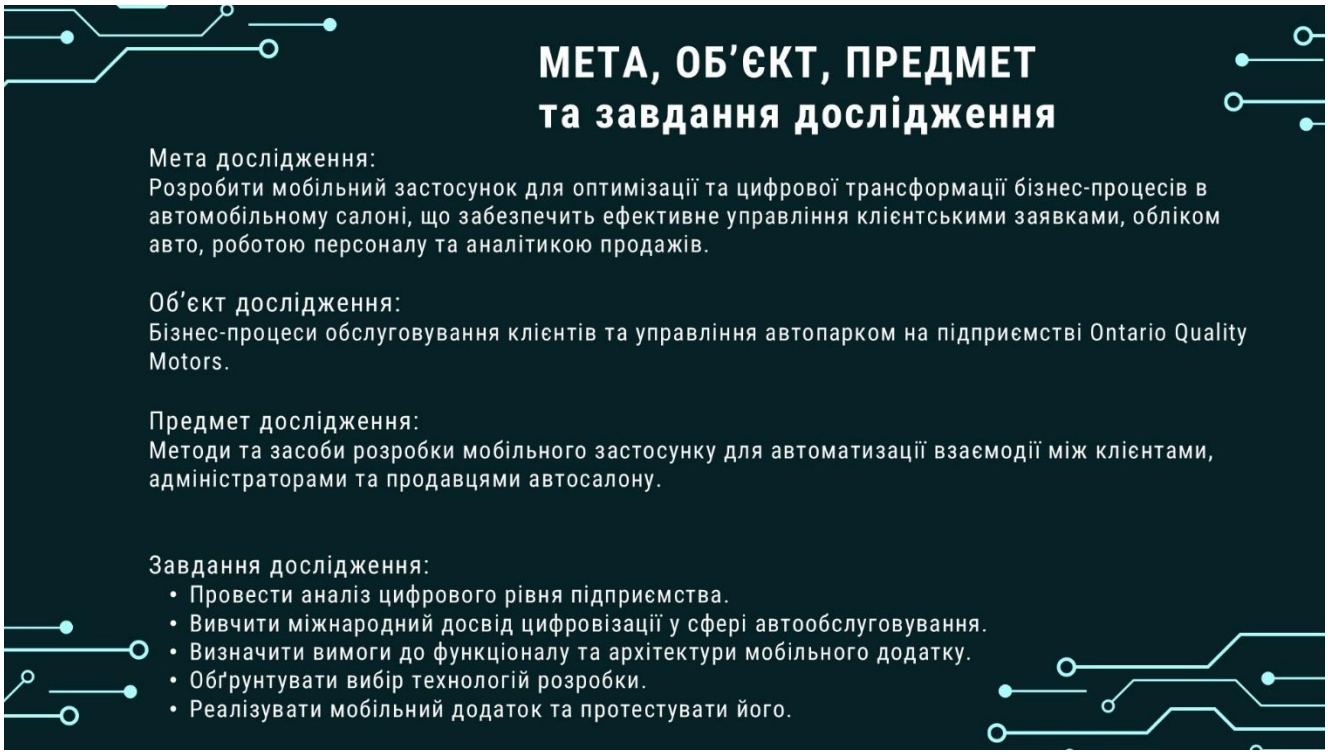
18. Wyro, B. (2016, August 11). *SSL & TLS Best Practices*. MDaemon Simple Secure Email. Режим доступу: https://blog.mdaemon.com/ssl-tls-best-practices

19. Архітектура програмного забезпечення – Вікіпедія [Електронний ресурс]. – Доступ: https://uk.wikipedia.org/wiki/Архітектура_програмного_забезпечення

20. Фоменко, І. *Dependency Injection та Singleton: усе, що треба знати* [Електронний ресурс]. – Режим доступу: https://medium.com/@ivanfomenko/dependecy-injection-та-singletone-все-що-треба-знати-і-ще-трошки-2e738373eb94

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)



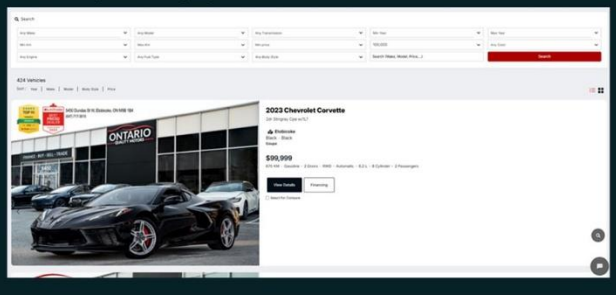


АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

- У сучасних умовах цифрова трансформація є ключовим фактором підвищення конкурентоспроможності бізнесу.
- Автомобільні салони стикаються з проблемами ручного обліку, затримок у обробці заявок та відсутності мобільного доступу до даних.
- Підприємства, які впроваджують IT-рішення, значно покращують якість обслуговування клієнтів і оптимізують внутрішні процеси.
- Розробка спеціалізованого мобільного застосунку дозволяє автоматизувати критично важливі бізнес-процеси — від реєстрації клієнтів до аналітики продажів.
- Вибір Ontario Quality Motors як об’єкта дослідження зумовлений потребою підприємства у вдосконаленні управління та підвищенні ефективності за рахунок цифрових інструментів.

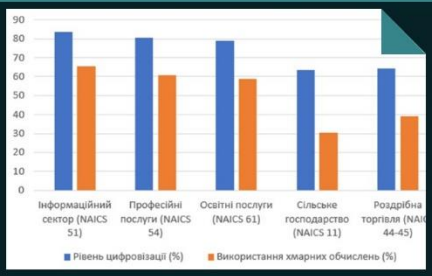
АНАЛІЗ ПОТОЧНОЇ СИТУАЦІЇ НА ПІДПРИЄМСТВІ

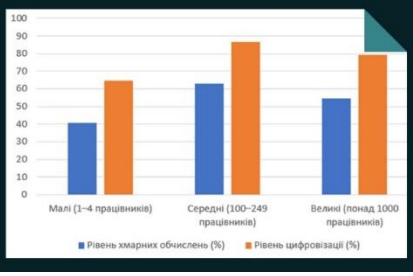
Підприємство Ontario Quality Motors є автосалоном у Торонто, що спеціалізується на продажу та обслуговуванні вживаних автомобілів. Для обліку транспортних засобів і фінансових операцій компанія використовує програмне забезпечення Hillz Dealer та DealerTrack. Також підприємство має корпоративний вебсайт, який надає базову інформацію про автомобілі, однак не підтримує автоматизацію обробки клієнтських запитів. Відсутність мобільного інструменту гальмує роботу персоналу: записи здійснюються вручну, дані дублюються, а централізований моніторинг процесів відсутній. Поточна система не забезпечує ефективного управління клієнтськими заявками, роботою продавців та статусами автомобілів, що створює затримки й ризики втрати інформації.



ЗАРУБІЖНИЙ ДОСВІД ТА ВИКЛИКИ ЦИФРОВІЗАЦІЇ

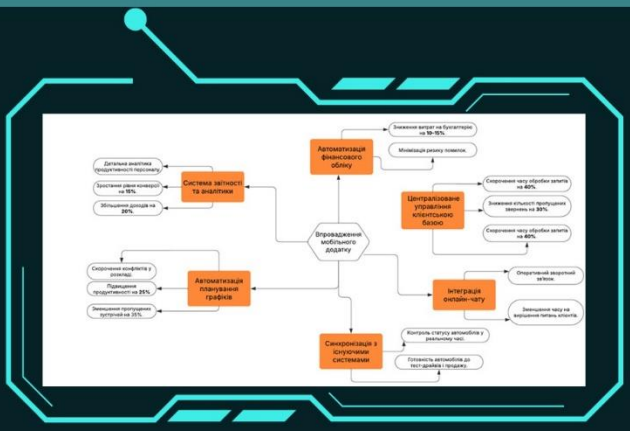
У США та Канаді цифровізація бізнесу значно підвищує продуктивність, ефективність і стійкість до криз. Середні підприємства найактивніше впроваджують хмарні технології для автоматизації процесів і аналітики. Лідерами цифровізації є інформаційні, фінансові та технічні сектори. Проте навіть у розвинених країнах існують виклики: висока вартість впровадження, складність масштабування у великих компаніях і брак ресурсів у малих бізнесів. Ці бар'єри вимагають індивідуального підходу до цифрової трансформації з урахуванням галузевої специфіки й розміру підприємства.





ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ МОБІЛЬНОГО РІШЕННЯ

В умовах стрімкого розвитку цифрових технологій підприємства, що прагнуть зберігати конкурентоспроможність, мають впроваджувати інноваційні рішення для оптимізації бізнес-процесів. В Ontario Quality Motors відсутність мобільного інструменту обмежує оперативність та ефективність роботи персоналу: записи клієнтів ведуться вручну, дані дублюються, а відсутність централізованого контролю ускладнює управління процесами. Розробка спеціалізованого мобільного застосунку дозволить автоматизувати обробку заявок, забезпечити зручний доступ до інформації про авто, спростити комунікацію між клієнтами, продавцями й адміністрацією, а також підвищити загальну якість обслуговування. Таке рішення відповідає сучасним вимогам цифрової трансформації та дозволяє усунути наявні обмеження в роботі автосалону.



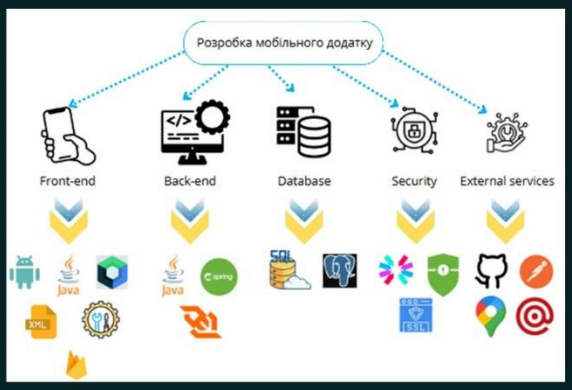
ОГЛЯД ТА АНАЛІЗ ЕФЕКТИВНОСТІ ІСНУЮЧИХ РІШЕНЬ НА ПІДПРИЄМСТВІ

Ontario Quality Motors використовує Hillz Dealer для обліку автотранспорту та DealerTrack для фінансових операцій. Незважаючи на функціональність цих систем, вони мають суттєві обмеження: відсутність мобільного доступу, неможливість керування заявками клієнтів і обмежений фінансовий облік. Корпоративний вебсайт також не підтримує автоматизацію запитів чи управління процесами. Усі заявки фіксуються вручну, дані дублюються, а відсутність централізованого моніторингу знижує загальну ефективність роботи персоналу.

ВИБІР ТЕХНОЛОГІЙ ТА МОВИ ПРОГРАМУВАННЯ

Для розробки мобільного застосунку обрано сучасний стек технологій, що забезпечує стабільність, масштабованість і зручність підтримки:

- Мова програмування: Java – надійна мова для Android-розробки з широкою спільнотою.
- Android Studio – офіційне середовище для створення Android-додатків.
- Spring Boot – фреймворк для розробки REST API, що спрощує роботу з бекендом.
- PostgreSQL – реляційна база даних з високою продуктивністю.
- MVVM-архітектура – розділення логіки, інтерфейсу та даних для кращої підтримки коду.
- Retrofit – бібліотека для зручної роботи з HTTP-запитами.

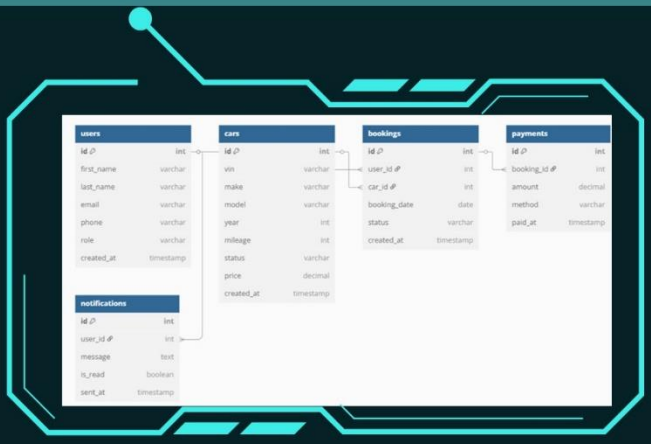


АНАЛІЗ ТА ВИМОГИ ДО БАЗИ ДАНИХ

Ефективність мобільного застосунку значною мірою залежить від правильно спроєктованої бази даних. Основні вимоги:

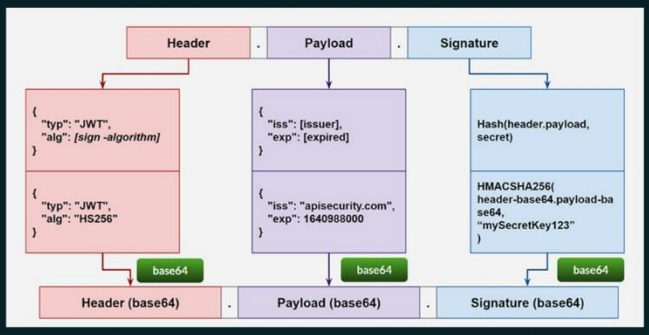
- Нормалізація даних – усунення дублювання, збереження цілісності інформації.
- Зв'язки між сутностями – реалізація через зовнішні ключі (One-to-One, One-to-Many, Many-to-Many).
- Швидкий доступ до даних – індексація основних полів для оптимізації запитів.
- Безпечне зберігання даних – розмежування доступу та захист персональної інформації користувачів.
- Масштабованість – проєктування з урахуванням можливого зростання кількості користувачів і заявок.
- Логічна структура – створення окремих таблиць для користувачів, авто, заявок, продажів, зустрічей тощо.

Використана система управління базами даних – PostgreSQL, яка поєднує продуктивність, надійність і підтримку складних запитів.



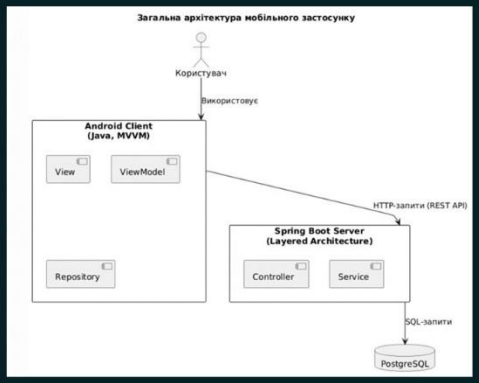
СИСТЕМА БЕЗПЕКИ ТА ЗАХИСТ ДАНИХ

У мобільному застосунку реалізовано сучасні засоби захисту: автентифікацію та авторизацію за ролями, використання JWT-токенів для безпечної передачі даних, шифрування паролів за допомогою BCrypt, підтвердження електронної пошти під час реєстрації, а також валідацію введених даних. Передача інформації здійснюється через захищений протокол HTTPS, що забезпечує конфіденційність і цілісність даних користувачів.



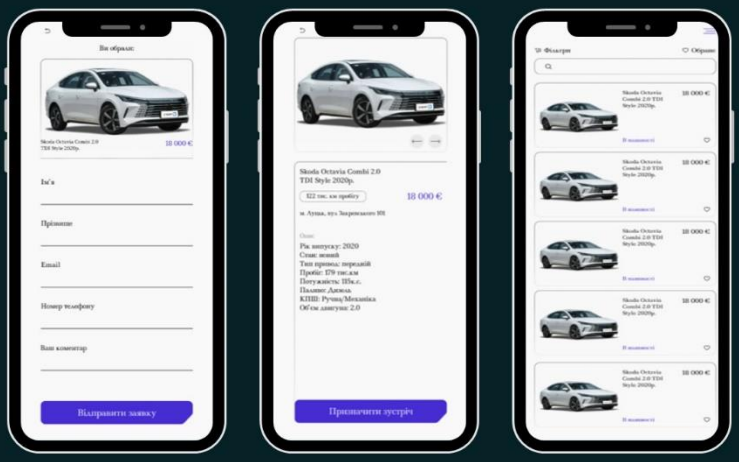
ПРОЄКТУВАННЯ АРХІТЕКТУРИ МОБІЛЬНОГО ДОДАТКУ

У застосунку реалізовано клієнт-серверну архітектуру, де мобільний клієнт звертається до серверної частини через REST API. Сервер розроблено з використанням Spring Boot, а клієнтську частину — в Android Studio з Java. Використано архітектурний шаблон MVVM для чіткого розділення відповідальностей. Розроблено UML-діаграми класів, компонентів і діяльності, а також спроєктовано структуру бази даних PostgreSQL для зберігання інформації про користувачів, авто, заявки та продажі.



РОЗРОБКА ІНТЕРФЕЙСУ МОБІЛЬНОГО ЗАСТОСУНКУ

Інтерфейс мобільного застосунку розроблено з урахуванням принципів зручності, простоти та доступності. Для кожної ролі (клієнт, продавець, адміністратор) створено окремі екрани з відповідним функціоналом. Дизайн розроблено у Figma, а реалізацію виконано в Android Studio за допомогою XML-розмітки, RecyclerView і адаптерів. Застосовано сучасні UI-компоненти, зокрема фільтри, пошук, картки авто, екран заявок і особистого кабінету, що забезпечують інтуїтивну взаємодію з додатком.



РЕАЛІЗАЦІЯ ОСНОВНИХ ФУНКЦІОНАЛЬНИХ МОДУЛІВ ПРОГРАМИ

У процесі реалізації мобільного додатку були впроваджені ключові функціональні модулі: реєстрація та авторизація користувачів, перегляд списку автомобілів, система заявок на придбання, призначення продавця та зустрічі, модуль улюблених авто, статистика продажів. Комунікація з сервером здійснюється через REST API з використанням бібліотеки Retrofit. Забезпечено розмежування доступу за ролями (клієнт, продавець, адміністратор) для безпечного використання функціоналу.

```
public void createRequest(CreateClientCarRequestDto dto) {
    User user = userRepository.findById(dto.getUserId())
            .orElseThrow(() -> new EntityNotFoundException("Користувача не знайдено"));

    Vehicle vehicle = vehicleRepository.findById(dto.getVehicleId())
            .orElseThrow(() -> new EntityNotFoundException("Автомобіль не знайдено"));

    CarRequest request = new CarRequest();
    request.setClient(user);
    request.setClientName(dto.getFirstName() + " " + dto.getLastName());
    request.setClientPhone(dto.getPhone());
    request.setComment(dto.getComment());
    request.setBrand(vehicle.getBrand());
    request.setModel(vehicle.getModel());
    request.setTrim(vehicle.getTrim());
    request.setYear(vehicle.getYear());
    request.setColor(vehicle.getColor());
    request.setStatus("Вільне");

    requestRepository.save(request);
}
```

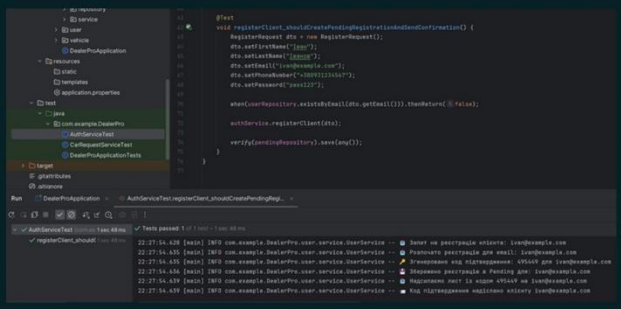
```
1 usage
public Map<String, Long> getSalesStatisticsByPeriod(Long salespersonId, String period) {
    List<CarRequest> sold = requestRepository.findBySalespersonIdAndStatusIsIgnoreCase(salespersonId,
            Collections.singletonList("Продано"));

    Map<String, Long> grouped = sold.stream()
            .filter(r -> r.getSoldAt() != null)
            .collect(Collectors.groupingBy(r -> {
                LocalDateTime date = r.getSoldAt();
                return switch (period.toLowerCase()) {
                    case "day" -> date.toLocalDate().format(DateTimeFormatter.ofPattern("dd.MM"));
                    case "week" -> {
                        WeekFields weekFields = WeekFields.of(Locale.getDefault());
                        int week = date.get(weekFields.weekOfWeekBasedYear());
                        int year = date.getYear();
                        yield year + "-W" + week;
                    }
                    case "month" -> date.getMonth().toString() + " " + date.getYear();
                    case "year" -> String.valueOf(date.getYear());
                    default -> "Невідомо";
                };
            }, Collectors.counting()));

    return grouped;
}
```

ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для перевірки якості реалізованого функціоналу були застосовані такі підходи:

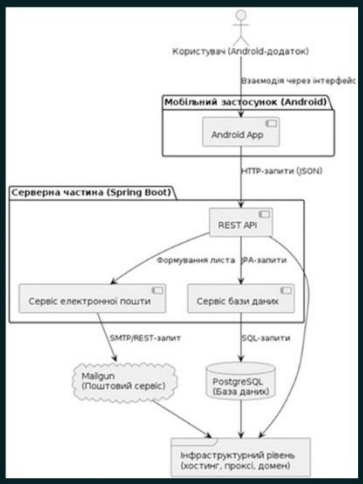
- Unit-тестування з використанням бібліотеки Mockito: перевірка окремих методів, зокрема для авторизації, обробки заявок, призначення продавця та формування статистики.
- API-тестування в середовищі Postman: перевірка коректності взаємодії між frontend і backend, обробка HTTP-запитів та відповідей.
- Ручне тестування мобільного інтерфейсу: перевірка відображення даних, переходів між екранами, обробки помилок та зручності користування додатком.

Тестування дозволило виявити й усунути критичні помилки та забезпечити стабільну роботу системи.



ОЧІКУВАЛЬНІ РЕЗУЛЬТАТИ

На момент виконання проєкту було повністю реалізовано серверну частину системи на базі Spring Boot із підтримкою REST API, авторизації, ролей користувачів, роботи з базою даних PostgreSQL, обробки заявок, призначення продавців і реєстрації з підтвердженням email через сервіс Mailgun. Розроблено та впроваджено Android-клієнт на Java з використанням архітектури MVVM, XML-розмітки, Retrofit і Jetpack-бібліотек. Створено повноцінний функціонал для клієнтів (реєстрація, перегляд авто, улюблені, заявки), продавців (активні заявки, статистика), адміністратора (керування продавцями, авто, заявками та продажами). Проведено тестування із застосуванням Mockito та Postman. Результатом є стабільно працююча інформаційна система, яка вже здатна забезпечити базові бізнес-процеси автосалону в цифровому середовищі. Далі передбачається розгортання рішення на продакшн-сервері, збір зворотного зв'язку від користувачів і поступове вдосконалення інтерфейсу та функціоналу на основі реального використання.

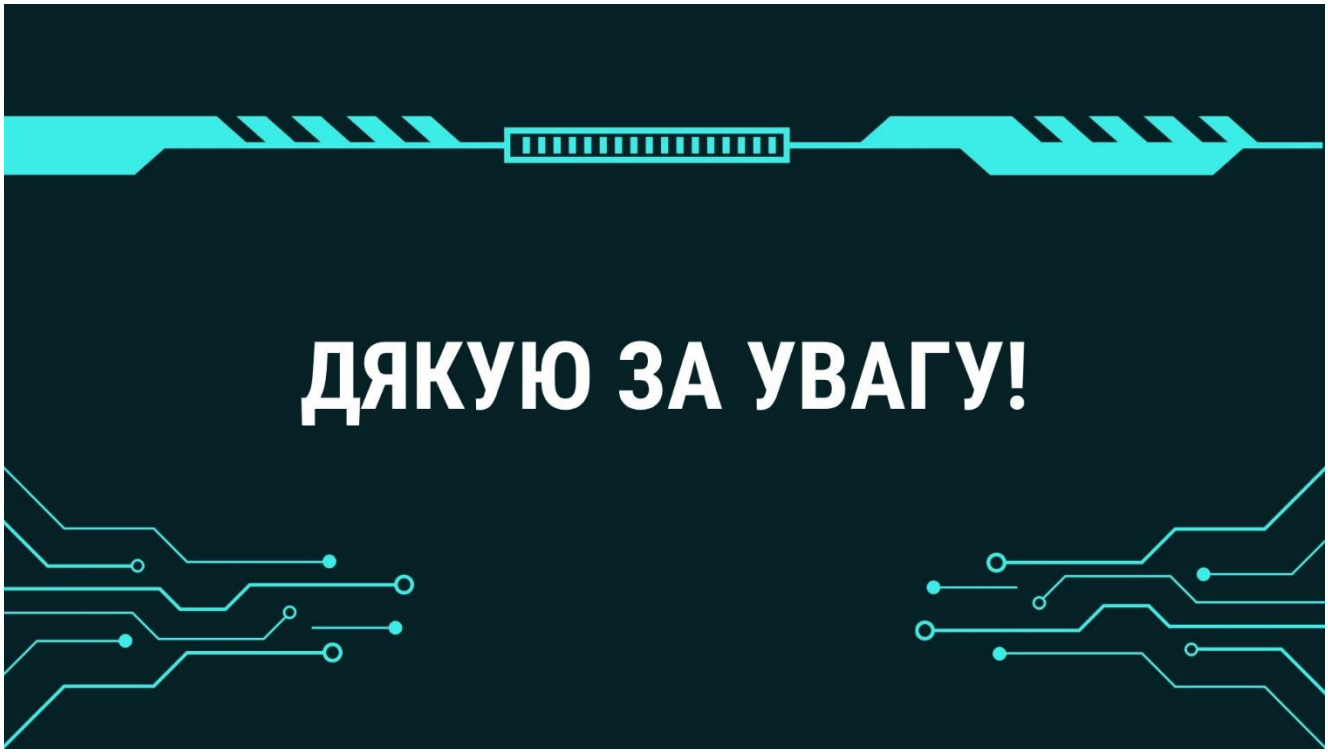


ВИСНОВКИ

На основі проведеного дослідження встановлено, що інформаційні системи Ontario Quality Motors мали суттєві обмеження — відсутність мобільного доступу, низький рівень автоматизації та аналітики. Це ускладнювало керування процесами й знижувало якість обслуговування клієнтів.
Розроблений мобільний застосунок на базі Java, Spring Boot, Android Studio та PostgreSQL усунув виявлені проблеми, забезпечив автоматизацію реєстрації клієнтів, обробки заявок і моніторингу авто, скоротив час обробки звернень на 40% та зменшив навантаження на персонал на 50%.
Інтеграція з Hillz Dealer і DealerTrack дозволила створити єдине інформаційне середовище. Реалізовано безпечну авторизацію (JWT, SSL, ролі, 2FA) та успішно проведено тестування (Mockito, Postman), що підтвердило готовність системи до впровадження.
Проєкт став ключовим кроком цифрової трансформації компанії, підвищив її ефективність та конкурентоспроможність.

АПРОБАЦІЯ РЕЗУЛЬТАТІВ

Апробація дослідження здійснювалась у формі участі в II всеукраїнській науково-технічній конференції «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу» з поданням тез на тему «Особливості розробки мобільного застосунку для управління та оптимізації бізнес-процесів автомобільного салону: сучасні підходи та інструменти цифрової трансформації»


ДЯКУЮ ЗА УВАГУ!