НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ПОГОДЖЕНО **Декан факультету**	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** **Завідувач кафедри комп'ютерних наук**
_____**Ігор БОЛБОТ** (підпис)	_____**Белла ГОЛУБ** (підпис)
«___» _____2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Інформаційна система обліку замовлень послуг таксі»

Спеціальність 122 – «Комп'ютерні науки»
Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми
д.е.н., професор _____ Руденський Р.А
(науковий ступінь та вчене звання) підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
ст. викладач _____ Міловідов Ю.О.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Вірянська Дар'я Василівна
(підпис) (ПІБ студента)

КИЇВ – 2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
____Комп'ютерних наук_____
__к.н.т., доцент__ _____ _Голуб Б.Л._
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"_" _____ 202 р.

ЗАВДАННЯ
на виконання бакалаврської кваліфікаційної роботи студенту
_____ Вірянській Дар'ї Василівні_____
(прізвище, ім'я, по батькові)

Спеціальність 122 – «Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи Інформаційна система обліку замовлень послуг таксі

Затверджена наказом ректора НУБіП України від "06" січня 2026р. №46 С

Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)

Вихідні дані до бакалаврської кваліфікаційної роботи

Розроблення необхідних модулів: реєстрація, авторизація та управління замовленнями.

Перелік питань, які потрібно розробити:

1. Аналіз предметної області
2. Проєктування інформаційного та програмного забезпечення
3. Розробка інформаційного та програмного забезпечення
4. Рекомендації щодо впровадження та експлуатації системи

Дата видачі завдання "30" вересня 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ Міловідов Ю.О.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ _Вірянська Д. В.
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ВСТУП 5
1 Системний аналіз предметної області 10
1.1 Постановка завдання....10
1.2 Огляд інформаційних джерел та існуючих рішень....11
1.3 Моделювання предметної області....22
2 Інформаційне забезпечення 27
2.1 Логічна модель даних....27
2.2 Вибір системи управління інформаційною базою....30
2.3 Створення інформаційної бази....33
3 Прикладне програмне забезпечення 38
3.1 Організаційна структура програмного забезпечення....38
3.2 Вибір інструментарію для створення прикладного програмного забезпечення....40
3.3 Алгоритмізація та програмування програмних модулів....42
4 Рекомендації щодо впровадження та експлуатації системи 46
4.1 Тестування системи....46
4.2 Вимоги до апаратного та програмного забезпечення....60
4.3 Склад інсталяційного пакету....62
ВИСНОВКИ 63
Список використаних джерел 65
Додаток А 66
Додаток Б 69
Додаток В 79

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІС – Інформаційна система.
ПЗ – Програмне забезпечення.
БД – База даних.
СУБД – Система управління базами даних.
SQL – Structured Query Language, мова запитів до баз даних.
Т/З – Транспортний засіб.
CVV – Тризначний код безпеки банківської карти.
API – Application Programming Interface, програмний інтерфейс.
C# – Мова програмування.
UML – Unified Modeling Language, мова моделювання.
AI – штучний інтелект.

ВСТУП

Саме сьогодні світ рухається ногою у сучасність. Трансформація суспільства, інформаційних технологій відіграють ключові ролі у підвищенні ефективності функціонувань підприємств. Особлива сфера трансформації інформаційних технологій відноситься і до транспортних послуг, де швидкість оброки інформації, точність виконання замовлень та якість є ключовими елементами на, що звертається увага. Умови, які створюються на сьогодні є високою динамікою приросту попиту, що потребує оперативно приймати рішення та ефективність організації взаємодії між клієнтами, водіями та диспетчерами.

Традиційні моделі організації роботи таксі зазначає частину процесів, що використовуються, як вручну або з використанням автоматизованих інструментів. Саме такі фактори призводять до появи низки проблем та недоліків у системі, що відзначається в оперативності обробки замовлення, дублювання інформацій або ж передачі даних між учасниками процесу. Дійсно це є великими недоліками, що негативно впливають на ефективність роботи підприємства, а найголовніше зацікавленість та задоволеність кінцевих користувачів.

З огляду на це виникає необхідність у створенні та прийняття ефективних рішень розробки, а саме у с творенні інформаційної системи обліку замовлення таксі, що дозволяє автоматизувати усі необхідні процеси. Основні критерії розробки, розробити та забезпечити централізоване збереження даних, оптимізація роботи диспетчерської служби. Впровадження подібної системи є необхідним, що дає змогу значно скорочувати час обробки, а найголовніше зменшити вплив людського фактора у системі та підвищити точність інформації.

Крім того, сучасні користувачі очікують необхідність від сервісів таксі не лише швидкого оформлення, але й прозорість процесів, можливість відстеження статусу поїздки, місцезнаходження тощо. У свою чергу адміністрації таких систем потребує більше інструментів аналізу діяльності, формування звітності та контролю ефективності роботи водіїв, що дає перевагу аналізувати та

приймати необхідні рішення в розробці або впроваджені у систему функціонал. Саме це формує потребу в комплексному програмному рішенні, що об’єднує всі етапи життєвого циклу замовлення в єдиній інформаційній системі.

Таким чином, розробка інформаційної системи обліку замовлень послуг таксі є актуальним завданням, що спрямовано на ефективне вирішення поставлених завдань під час проектування. Підвищення ефективності управління транспортними послугами, покращення якості обслуговування клієнтів з оптимізацією внутрішніх бізнес – процесів у підприємстві.

Актуальність

Актуальність розробки ІС обліку замовлень послуг таксі зумовлена на стрімкий розвиток цифрових технологій та підвищення вимог до якості сервісів у сфері транспортних перевезень. Сучасність служб таксі функціонують в умовах великої конкуренції, де швидкість реагування на запити, впровадження нових технологій – це головним фактором саме такого бізнесу.

Практика відзначає, що є певні транспортні компанії з використанням автоматизованих або застарілих систем обліку замовлення, не дозволяючи у повній мірі забезпечувати оперативність, актуальність виконання замовлень тощо.

Важлива проблема – недостатній рівень інтеграції між учасниками процесу. Відсутність єдиного інформаційного середовища ускладнює взаємодію між клієнтами, диспетчерами та водіями, що звісно негативно впливає на загальну ефективність роботи системи.

У зв’язку з цим виникає необхідність створення саме сучасної ІС, що дозволить автоматизувати процеси обліку замовлень, забезпечити централізоване зберігання даних та підвищити швидкість оброки інформації. Зменшення впливу людського фактору значно зменшиться в рази та підвищить точність даних та стабільну роботу транспортного підприємства.

Тема

Тема бакалаврської кваліфікаційної роботи полягає у розробці ІС обліку замовлень послуг таксі, що забезпечуватиме автоматизацію основних процесів та діяльності служби перевезень.

Саме обрана тема орієнтована на створення єдиного інформаційного середовища для взаємодії між клієнтом, водієм та адміністративним персоналом. Виконання повного циклу оброки замовлення від початку його створення до завершення поїздки з фіксацією результатів у базі даних.

Також увага приділена побудові структурованій моделі користувачів з різними рівнями доступу, що дозволить забезпечувати безпеку даних та чіткий розподіл функціоналу.

Зв'язок роботи з науковими програмами, планами, темами

Виконання бакалаврської кваліфікаційної роботи здійснюється відповідно до напрямів наукових досліджень у сфері інформаційних технологій та автоматизованих систем управління.

Робота узгоджується з сучасним підходом до цифрової трансформації бізнес – процесів у транспортній галузі та спрямована на розробку прикладного ПЗ з підвищення ефективності.

Процес виконання роботи використовувало принципи системного аналізу, ООП проєктування та моделювання інформаційних процесів.

Об’єкт дослідження

Об’єктом досліджень є окремі процеси організації з управлінням замовлень у службах таксі, з інформаційним потоком, що виникають у процесі взаємодії між клієнтами, водіями та диспетчерською системою.

Розглядаються, як традиційні способи вирішення обробки замовлення, так і сучасні цифрові платформи, що використовуються для автоматизації транспортних послуг. Особлива увага буде приділена структурі інформаційних взаємодій та способом збереження даних у системі.

Предмет дослідження

Предмет дослідження є метод та засоби саме проєктування з реалізацією ІС. До предметної області належатиме алгоритми обробки, структура даних,

механізм взаємодій, а також принцип організації. Окремо багаторівневої системи доступу, що розмежувати права користувачів відповідно до їх ролей.

Задачі дослідження

1. Аналіз існуючи інформаційних систем у сфері таксі та визначити їх переваги та недоліки.
2. Формування функціональності до розроблювальної системи.
3. Проєктування логічної та фізичної структури БД.
4. Реалізування ІС з використанням мови програмування C# у середовищі Windows Forms.
5. Розроблення необхідних модулів, а саме: реєстрація, авторизація та управління замовленнями.
6. Механізм розподілу ролей користувачі з рівнями доступу до системи.
7. Формування звітної інформації для адміністратора системи, а навіть для диспетчера(менеджера).
8. Проведення тестування ПЗ та оцінити стабільність розроблювальної системи.

Методи дослідження

Процес розробки ІС використовував комплекс наукових та більше інженерних методів дослідження. Основним є методом системного аналізу, що дозволяє дослідити структуру предметної області та визначити взаємозв’язки між компонентами.

Формалізація процесів застосовувалась за допомогою UML – діаграм, що дозволяє візуалізувати сценарії роботи та логіку взаємодій. Проєктування БД виконувалося на основі реляційного підходу з використанням нормалізації, що забезпечило цілісність та коректність збереження даних. Програмна реалізація базувалася на основі ООП підходу, що дозволило забезпечити модульність.

Теоретична значимість

Теоретична значимість роботи полягає в тому, що потрібно провести дослідження принципів побудови ІС для автоматизації певних процесів. У межах

дослідження було розглянуто підходи до організації реляційних БД, принципи моделювання ІС, а також саме головне методи побудови програмної архітектури. Результат може бути, як теоретична база для подальших досліджень у сфері автоматизації.

Практична значимість

Практична значимість розроблювальної ІС полягатиме у можливості її застосування для автоматизації усіх необхідних процесів з оптимізацією процесів обробки даних.

Саме це дозволятиме забезпечити централізоване управління даних, оперативну обробку та контроль в режимі реального часу. Додатково механізм розмежування прав доступу, що дійсно підвищує рівень безпеки та організацію роботи користувачів системи.

Впровадження даного програмного рішення сприяє підвищення ефективності роботи підприємства та покращенню якості обслуговування клієнтів.

1 Системний аналіз предметної області

1.1 Постановка завдання

Розробка ІС обліку замовлень послуг таксі, має актуальність у повсякденному житті та маючи попит на транспортні послуги, які надаються різноманітними компаніями. Сучасність надає нам можливість розвиватись, що не скажеш про інші компанії. Сучасні служби таксі стикаються з проблемами, які зв’язані з ручними обробками замовлень, некоректним розподілом водіїв, втратою даних про клієнтів та відсутність прозорості у відстеженні поїздок, що призводить до зниження якості обслуговування та збільшення операційних витрат.

Метою даної роботи є проєктування та реалізація ІС, що дозволить пришвидшити та автоматизувати процес обліку замовлень послуг таксі, та забезпечити між трьома ключовими учасниками процесу, а саме: клієнт, менеджер та водій. Кожен з учасників має свою роль в системі, задачі та функціональність, що насамперед дозволяє аналізувати, керувати та проводити роботу над помилками.

Клієнт має змогу реєструватись, заповнювати особисті дані, та оформляти замовлення, як за допомогою диспетчера так і самостійно. Саме наповненість даних дає привілеї для того, щоб водій бачив усі необхідні дані, які необхідні йому. Менеджер, а саме диспетчер, у свою чергу здійснює обробку вхідних замовлень та призначає водія за потреби. А саме водій, працюючи з мобільного телефону, використовуючи вже застосунок де одразу вибирається замовлення та приймається, вже вбудовано побудова маршруту від місце знаходження водія до пункту подачі авто, та від місця подачі до місця пункту призначення. Це дає змогу використовувати наш застосунок по повній.

Система реалізована у вигляді двох взаємопов’язаних компонентів: десктопний застосунок для менеджера, що розроблена на Windows Forms та мобільний застосунок, який розроблений за допомогою .NET MAUI. В якості управління базами даних, використовується PostgreSQL, що забезпечує надійне зберігання та цілісність даних.

Практична цінність розробленої системи полягає у скороченні часу оброки замовлення, мінімізація людського фактору при розподілі водіїв та підвищення загальної якості транспортного обслуговування. Впровадження подібного рішення дозволяє службам таксі перейти від ручного управління до повноцінного процесу обліку замовлення.

1.2 Огляд інформаційних джерел та існуючих рішень

Аналіз існуючих рішень у сфері ІС обліку замовлення послуг таксі є важливим та невід'ємним етапом у процесі розробки системи. Саме такий аудит дозволяє з одного боку, ознайомитись з технологіями та підходами, що саме застосовуються у предметній області, а саме з іншого боку – виявляти переваги та недоліки наявних рішень з метою формування вимог до власної розробки ІС. Для порівняльного аналізу було прийнято рішення обрати три сервіси, а саме представники ринку: DLS Trans, Bolt та Uklon.

На рис. 1 та рис. 2 зображено сервіс транспортного обслуговування DLS Trans, що надають послуги таксі, а саме за безготівковими рахункам.

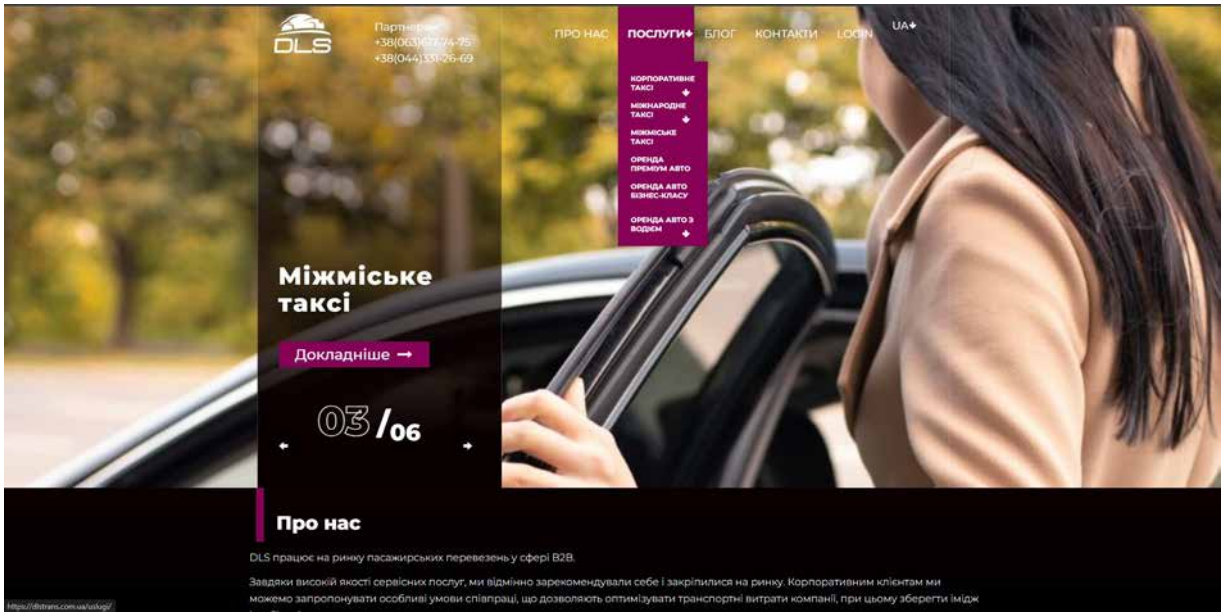

Рис. 1 Електронний сервіс надання послуг транспортного обслуговування.

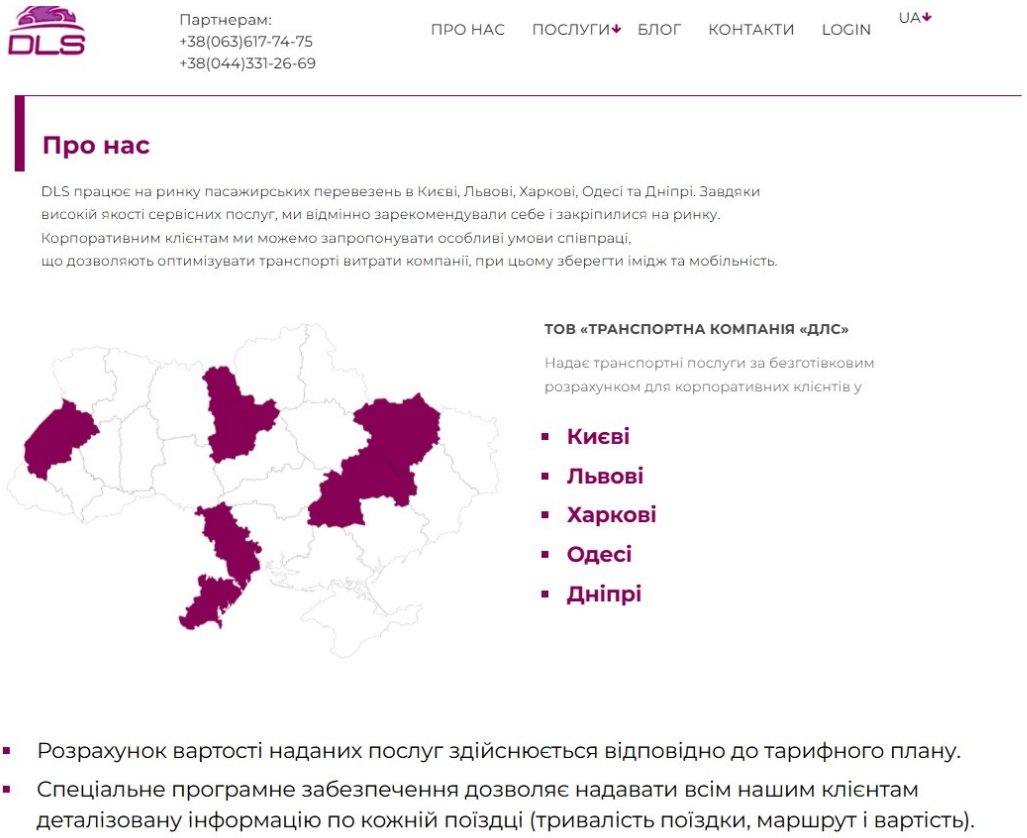

Рис. 2 Інформація про їхній сервіс.

DLS Trans – транспортна компанія, що спеціалізується на наданні послуг корпоративного таксі, міжнародних та міжміських перевезень у сегменті B2B. Компанія орієнтована на корпоративних клієнтів, що пропонує їм централізований облік поїздок, статистику по поїздкам та особливі персоналізовані умови співпраці. Серед ключових послуг виділяються такі перевезення, як: оренда автомобілів преміум та бізнес – класу, групові пасажирські перевезення, а також міжнародні трансфери.

Організація використовують процес оброки замовлень за допомоги диспетчерської моделі прийому замовлень. В основному замовлення приймаються через додаток Telegram, де присутні диспетчера, клієнт надає інформацію у вигляді кількості пасажирів, адресів та контактний номер одного із пасажирів або декількох. Також приймаються замовлення через додаток або веб-сайт безпосередньо по телефону через оператора. Це наближає модель роботи компанії до функціональності розроблювальної системи, так, як це

передбачає активну участь менеджера у процесі призначення водія та координації поїздки.

Платформа підтримує безготівковий розрахунок клієнтів компанії, що ж перевагою для корпоративних клієнтів, надається єдиний консолідований рахунок за всі поїздки передаючи звітність за місяць використання сервісом. Також є одним із найголовніших переваг – це система забезпечує збереження детальної інформації по кожній поїздці із зазначенням часу, місцем подачі та висадки, відстань, водіїв та вартості.

Серед недоліків хотілось би підкреслити відсутність публічного доступного програмного інтерфейсу для інтеграцій з іншими системи, а також обмежену прозорість щодо внутрішньої архітектури платформи управління замовленнями. Присутній лише мобільний додаток для водіїв, що має обмежену можливість користування, а саме система забезпечує тільки режим навігації для маршрутів, що звісно поступається міжнародним аналогам.

Додатково слід зазначити, що модель роботи компанії DLS Trans значною мірою використовує в повному обсязі та звісно, що залежить від людського фактору, оскільки процес розподілу замовлень між водіями здійснюється виключно диспетчером вручну або напівавтоматизовано. Що звісно завдає певні обмеження у швидкості прийняття рішень, особливою увагою в одночасності декількох підряд замовлень. Такий рівень сервісу, конкретно впливає на швидкість та якість обслуговування клієнтів.

Варто відзначити, компанія забезпечує досить високий рівень контролю та обліку замовлень поїздок, існуюча система не завжди дозволяє ефективно та аналітично обробити дані в режимі реального часу. Формування здійснюється виключно за період, що також обмежує швидкість надання інформації.

Важливим аспектом є недостатній рівень автоматизації процесів, взаємодії тощо. Комунікація між клієнтом зазначається виключено через диспетчера транспортної служби, якою дійсно може бути проблема, якщо комунікація йде наперед з клієнтом та диспетчером? А саме проблема складається в тому, що може трапитись втрата або передача некоректних та необхідних даних для

замовлення. Це також є чинником ускладнення централізованого збереження всієї історії взаємодії.

З технічної точки зору, платформа компанії має потенціал для подальшого розвитку, зокрема впровадження API – рішень для інтеграції з зовнішніми сервісами. Відсутність відкритого програмного інтерфейсу обмежує можливість масштабування системи та адаптації під інші бізнес – моделі.

Окремо підкреслюючи, що наявність мобільного застосунку для водіїв є позитивним фактором, однак його мала ефективність у процесах здебільше зменшу притягнутість клієнтів або ж навіть водіїв до майбутнього використання. Розширення функціоналу могло б включати управляння статусами, обмін повідомлень, доступ до аналітики власних поїздок. На мою думку саме це б підвищило ефективність роботи водіїв та власне майбутній рівень інтеграції.

Таким чином, аналізуючи діяльність DLS Trans дозволяє зробити висновок те, що попри високий рівень організації сервісу та орієнтації, існуюча модель має низку обмежень пов’язаних з автоматизацією, інтеграцією та аналітичною підтримкою. Такі параметри потрібно враховувати при проєктуванні нової ІС, що спрямована на підвищення ефективності обробки замовлення та оптимізацією.

Наступний на черзі буде додаток BOLT. На рис. 3 зображено додаток сервісу транспортного обслуговування BOLT, а саме інтерфейс мобільного застосунку Bolt.

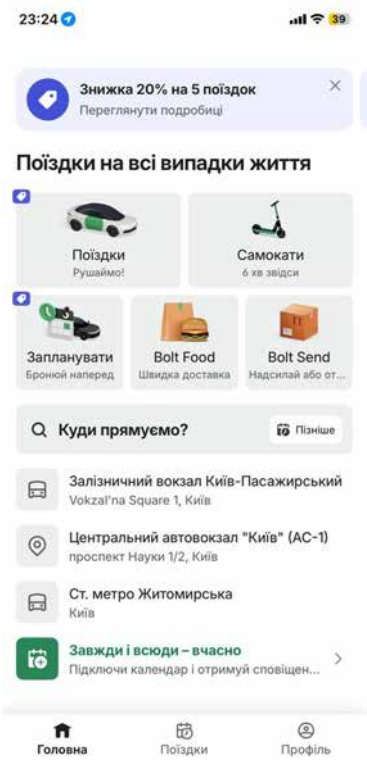

Рис. 3 Інтерфейс мобільного застосунку Bolt.

Bolt – один із провідних міжнародних сервісів замовлення транспортних послуг, що використовується широко на території України та країн Європи. Платформа реалізована у вигляді двох окремих мобільних застосунків, а саме: для клієнта та водія, що йде до централізованої серверної інфраструктури що обробляють запити в реальному часі.

Процес оформлення замовлення в застосунку Bolt відбувається наступним чином:

1) Клієнт відкриває застосунок, що за його дозволом дозволяє визначати поточне місцезнаходження за допомогою GPS;
2) Наступний процес, клієнт вказує адресу подачі авто та обирає собі клас авто: стандарт, бізнес, преміум, комфорт;
3) Система автоматично розраховує вартість поїздки;
4) Після підтвердження замовлення алгоритм платформи автоматично підбирає авто клієнту, що відверто кажучі буде вже надалі залежити від водія;

5) Після прийняття замовлення, клієнт вже бачить усю необхідну інформацію для клієнта: інформація хто везе, авто, номер авто тощо;
6) Вже під час завершення поїздки, клієнт може розрахуватись готівку, як обрав на початку замовлення, або з нього вже списали гроші з карти, під час підтвердження водієм, що водій дійсно береться за це замовлення.

До переваг платформи Bolt, відноситься: висока автоматизація під час усього процесу замовлення, вбудована навігація для водія, рейтинг водіїв та клієнтів, підтримкою готівкової та безготівкової форми сплати, широке географічне охоплення, а саме, як було описано зверху, більшість країн Європи, а також прозора система динамічного ціноутворення.

Суттєвими недоліками системи з точки зору задачі є – повноцінна відсутності ролі диспетчера або можливо менеджера, який, як учасник процесу обробки замовлення. Вся логіка падає на розподіл замовлення між водіями, що дає змогу автоматично виконувати всю роботу без диспетчера або менеджера, а це є великою проблемою для малого бізнеса, що не має власної диспетчерського центру, аналітики тощо. Платформа не надає інструментів для ведення обліку поїздок для компаній за потреби.

Слід зазначити, що архітектура платформи Bolt базується на високорівневій автоматизації процесів, що реалізується за допомогою алгоритмів динамічного підбору водіїв. Саме такий підхід забезпечує мінімізувати час очікування для клієнта та оптимізацію маршруту. Однак потрібно розуміти, подібна модель функціонування є жорстко централізованою та закритою, що обмежує можливість зовнішнього втручання або адаптації.

Важливий аспект є те, що система орієнтована переважно на масовий ринок використання сервісом, у зв’язку з чим присутнє певне обмеження, а саме функціонал корпоративного управління. Платформа представляє доступ до бізнес – акаунтів, але вона не забезпечує повноцінного інструментарію для централізованого обліку поїздок, формування звітності або інтеграції з іншими

внутрішніми системами. Звісно це буде створювати труднощі для компанії, які потребуватимуть детального контролю витрат.

Зазначимо, що система представляє собою відсутність ролі диспетчера або адміністративного оператора, у класичному розумінні зміню саму модель взаємодії. Усі призначення водія приймаються автоматизованим алгоритмом. Прикладом буде те, що не можна вручну скоригувати розподіл замовлення або врахувати додаткові бізнес – вимог клієнт безпосередньо через оператора.

З точки зору ІС також слід зазначити, що доступ внутрішньої аналітики платформи є суттєво обмеженим. Користувачі мають лише базову інформацію про свої поїздки, тоді, як розширена статистика алгоритму розрахунку вартості або параметри розподілу залишаються невід'ємно закритими. Звісно таким чином це унеможливлює детальний аналіз ефективності використання сервісу.

Окрема увага потребує питанню інтеграційних можливостей. Платформа Bolt практично не надає відкритого API для сторонніх розробників або корпоративних систем, що звісно обмежує використання у комплексних інформаційних рішень. У сучасності такі умови розвитку екосистеми може розглядатись, як фактор, що зменшує гнучкість масштабування.

Система демонструє високий рівень надійності та стабільності, вона орієнтована на переважно універсальний сценарій використання, що звісно не завжди враховує специфіку локальних ринків. Саме завдяки цьому – це створює простір для майбутніх розробок альтернативних рішень, що звісно будуть поєднувати автоматизацію процесів з можливістю ручного управління та адаптації.

Таким чином, аналіз платформи Bolt показує, що попри високі технології, реалізація та автоматизація для кінцевого користувача має низку обмежень у сфері корпоративного управління, аналітики та інтеграції, що підтверджує доцільність розробки спеціалізованої ІС обліку замовлень послуг таксі, що буде поєднувати усі ці процеси.

Наступним та останнім буде застосунок, що на мою думку є – гігант ринку в сфері таксі, а саме – Uklon. На рисунку 4 буде зображено інтерфейс мобільного застосунку.

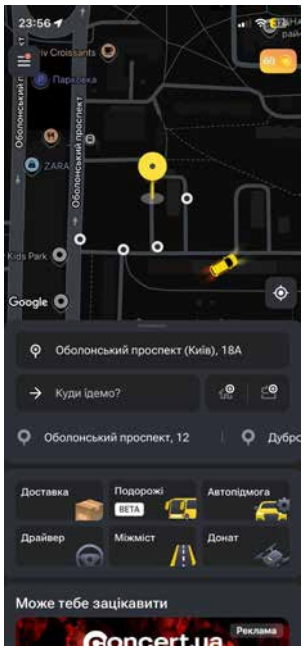

Рис. 4 Інтерфейс мобільного застосунку.

Uklon – це розробка, де збираються усі ключові спеціалісти та роблять переворот IT індустрії, та працюють на заохоплення та масштабування сервісу. Прикладом буде зображено їх програми лояльності, що буде зображено на рис. 5.

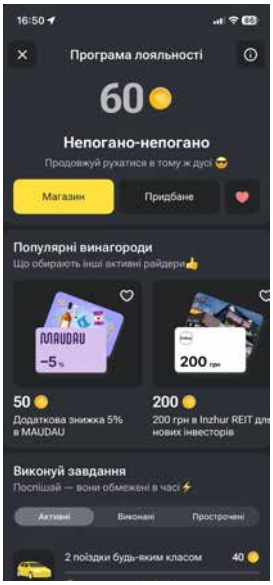

Рис. 5 Програма лояльності запроваджено фахівцями у 2021 році.

Uklon – це українська розробка, а саме платформа замовлення таксі, послуг транспортування клієнтів тощо. Саме те, що враховує специфіку вітчизняного ринку та користується значним попитом серед населення України. Велика різниця стосовно міжнародних клієнтів – це те, що сервіс підтримує можливість замовлень через оператора, що наближає його функціональність до розроблювальної системи, а саме інформаційна система обліку замовлень таксі.

Етап оформлення замовлення у застосунку Uklon, робиться наступним чином:

- Клієнт реєструється у застосунку, вказуючи власний номер телефону та особисті дані користувача;
- При оформленні замовлення клієнт вказує адресу відправлення та призначення обираючи необхідну категорію та зручний для нього сплату за поїздку;
- В свою чергу система розраховує вартість вартість поїздки на основі поточного тарифу та відстані маршруту від точки поскадки та висадки клієнта;
- Клієнт може відстежувати де знаходиться авто перед подачею автомобіля, та під час поїздки, і це проходить все в режимі реального часу;
- Після вже завершення поїздки клієнт має можливість оцінити якість поїздки та обслуговування, за потреби можливо залишити чайові, а водій в свою чергу оцінює клієнта та також виставляє рейтинг користувачу, для того, щоб у майбутньому всі інші водії бачили цей рейтинг у користувача.

На рис. 5 можемо побачити демонстрацію оформлення замовлення, що показує процес оформлення замовлення клієнтом послугу.

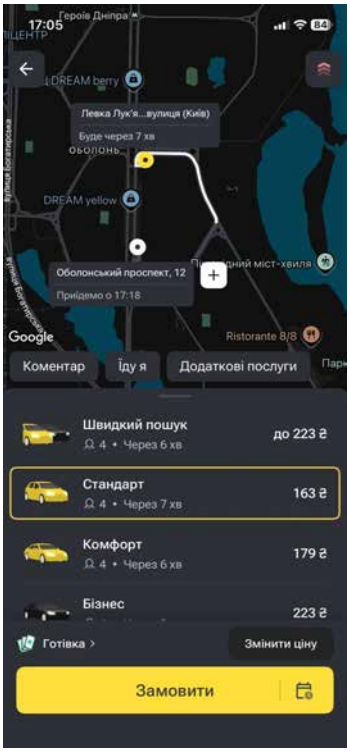

Рис. 6 Інтерфейс процесу замовлення послуг транспортування, а саме таксі.

Серед переваг компанії, варто підкреслити та зазначити – підтримка готівкового розрахунку сплати за послуги транспортування, що при цьому маємо динамічну систему ціноутворення з можливістю вибору категорії автомобіля, наявністю веб-інтерфейсу для управління автопарком, а також орієнтацію на український ринок з урахування його специфікації та вибору замовлень і його особливостей.

Стосовно недоліків платформи – це те, що вона орієнтована на велику кількість масового споживчого ринку і не дає достатніх інструментів для індивідуального налаштування потреб конкретних служб таксі з власним диспетчерським центром. Крім того, функціональність десктопного інтерфейсу для менеджера є з великим обмеженням та не дозволяє повноцінно керувати процесом призначення водіїв вручну.

Щоб зрозуміти всю картину, яка представлена мною всі проєкти буде створена порівняльна таблиця 1 існуючих рішень, щоб побачити та підвести підсумки переваг та недоліків систем.

Таблиця 1

Порівняння існуючих рішень

Сервіс	**Переваги**	**Недоліки**
DLS Trans	Система має розподілення між ролями користувачів, розподілення функціональності тощо; Роль менеджера або диспетчера; Мається мобільний застосунок для водія; Часткова навігація реального часу. Ручне призначення водія; Корпоративний облік поїздок та велика перевага – це адаптація під мали бізнес.	Відсутність десктопного застосунку, що зменшує мобільність компанії; Підтримка готівкової оплати відсутня, що зменшує клієнтоорієнтованість компанії.
Bolt	Наявність мобільного застосунку для водія та клієнтів, що є основною суттю застосунку; Навігація працює у реальному режимі часу; Підтримка готівкової сплати під час поїздки.	Відсутність розподільчої системи для диспетчера або менеджера; Ручне призначення; Десктопний інтерфейс менеджера; Корпоративний облік поїздок; Адаптація під малий бізнес.
Uklon	Система має розподілення функціональності диспетчера; Мобільний застосунок, як для водія так і окремий додаток для клієнта; Навігація в режимі реального часу; Ручне призначення водія; Десктопний інтерфейс менеджера; Підтримка готівкової сплати за замовлення; Адаптація під малий бізнес.	Корпоративний облік поїздок.

1.3 Моделювання предметної області

Під час моделювання предметної області було прийнято рішення уніфікувати мову моделювання UML, що відіграє ключові ролі під час проектування ІС. Моделювання предметної області дозволяє зрозуміти логіку функціонування розроблювальної системи обліку замовлення послуг таксі, що в майбутньому дозволяє виявити основні компоненти, ролі користувачів та взаємодія між ними.

У межах цього дипломного проєкту моделювання використовується для опису процесів оформлення замовлення, керуванням транспортним засобом, адміністрування та організація взаємодій між клієнтом, диспетчером та водієм.

Розроблювальна система включає декілька програмних компонентів. Мобільний застосунок, що створений використання технологій .NET MAUI, призначений для клієнтів та водіїв, що має змогу за допомогою мобільного застосунку створювати клієнтом замовлення, обирати клас та спосіб сплати, а водій зможе отримувати всю необхідну інформацію про замовлення та взаємодіяти цією інформацією в режимі реального часу. Окремою реалізацією відзначається проєктування десктопного застосунку для диспетчера, що реалізовано на платформі Windows Forms, основним призначенням є обробка замовлення, контроль доступного транспорту та координація процесу перевезення та корпоративних підприємств.

Опису поведінки системи та її функціональних можливостей було прийнято рішення використати UML – діаграми поведінки, зокрема:

1) діаграму прецедентів;
2) діаграму активності;
3) діаграму послідовності.

Діаграма прецедентів головний інструмент аналізу для проєкту, а саме вимоги до ПЗ, оскільки це все відображає взаємодію між акторами та системою, демонструючи перелік доступних функцій. Саме така діаграма дозволяє описати бізнес-логіку ІС та визначити межі відповідальності кожного користувача.

Для розробки було прийнято рішення розробити мовою моделювання використовуючи UML діаграми поведінки: діаграма прецедентів, активності та послідовності.

На рисунку 7 буде відображено діаграму прецедентів, що відображає розроблювальну систему служби таксі.

Візуальне моделювання в UML можна представити як деякий процес порівневого спуску від найбільш узагальнений і абстрактній концептуальній моделі початкової системи до логічної, а потім і до фізичної моделі відповідної програмної системи. Для досягнення цих цілей спочатку будується модель у формі так званої діаграми варіантів використання (use case diagram), яка описує функціональне призначення системи або, іншими словами, те, що система робитиме в процесі свого функціонування. Діаграма варіантів використання є початковим концептуальним представленням або концептуальною моделлю системи в процесі її проектування і розробки.[1]

Розроблювальна діаграма представляє з себе наявність 3 акторів, а саме головних акторів цієї системи: користувач, менеджер, адміністратор. Система також має не тільки акторів, а і прецеденти: оформлення замовлення таксі, що має тип сплати та перегляд запропонованих варіантів класу авто, пошук вільних автівок, оформлення замовлення, та більш адмініструвальні дії, робота з даними узагальнено.

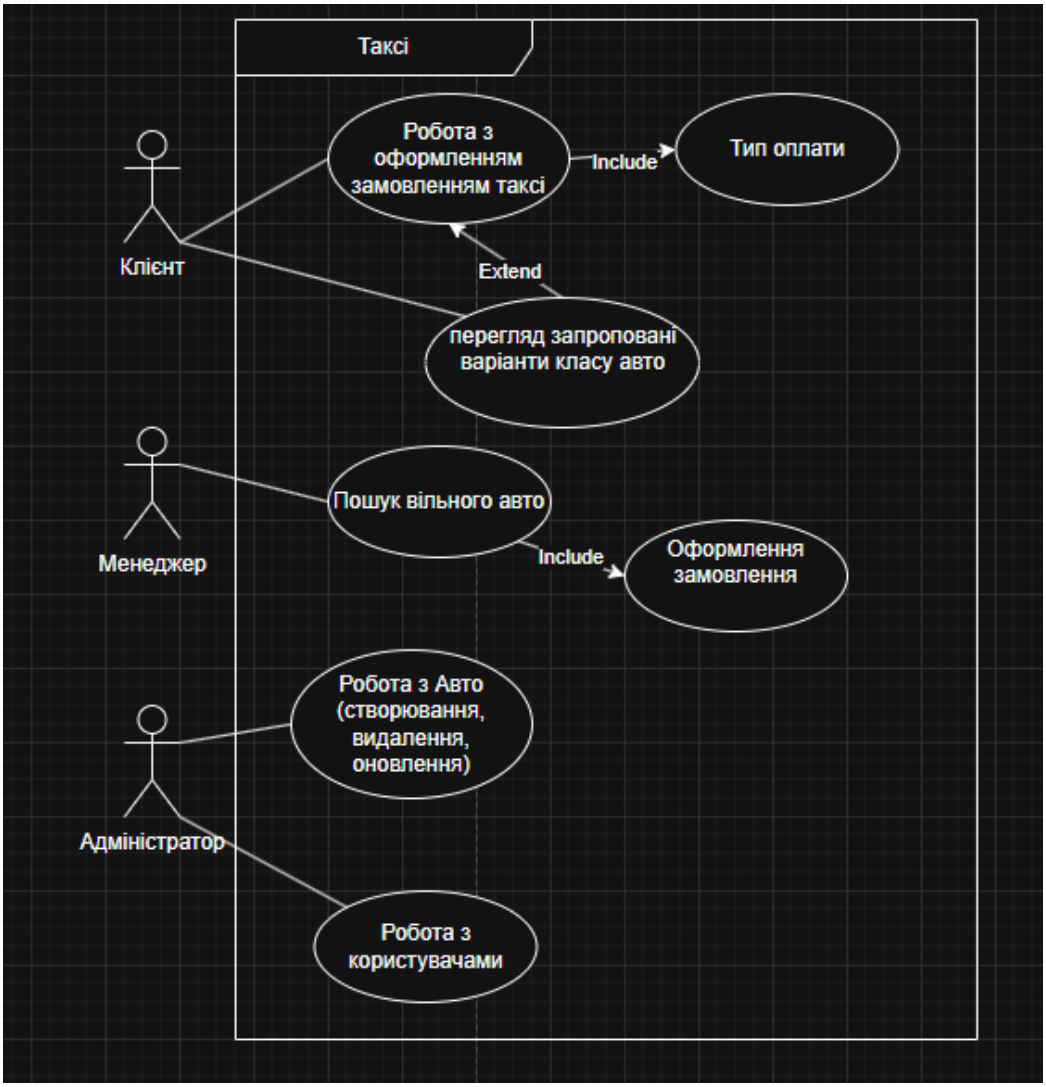

Рис. 7 Діаграма прецедентів

1. **Клієнт (користувач):**

Основний актор системи, так, як він користується послугами таксі та ініціює створення замовлення. Клієнт має можливість оформити замовлення на поїздку, вказавши необхідні параметри, а саме тип класу автомобіля, місце подачі та кінцевий пункт призначення. Це однозначно дає можливості швидко та зручно оформити замовлення, тому, що клієнту важливо швидко, зручно та зрозуміло зробити своє замовлення.

2. **Менеджер:**

Менеджер, а саме диспетчер відповідає за роботу з корпоративними компаніями, а саме кожна компанія зможе мати власного менеджера, який буде відповідати за той чи інший напрямок(розподілення по районам). Менеджер

безпосередньо буде надавати звітність компаніями, призначати водіїв, а також додатково, клієнти зможуть звертатись до нашої компанії безпосередньо викликати авто, щоб не робити зайвих рухів, якщо клієнту терміново потребується авто.

3. **Адміністратор:**

Адміністратор відповідає за працездатність системи та адміністрування її даних. До його функціональності в системі відноситься управління обліковими записами в системі, робота з інформацією даних про транспортні засоби. Адміністратор зможе створювати за потребою та редагувати з видаленням даних про автомобіль, що дозволяє підтримувати актуальність системи та забезпечувати коректну роботу системи.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних

Модель даних – це засіб для визначення логічного зображення фізичних даних, що відносяться до деякого додатку. В процесі розробки засад створення баз даних вибір моделі в основному залежить від об'єкту впровадження (від регіонального представництва до центрального органу управління), а також від етапу впровадження.

Модель «сутність-зв'язок» (ER-модель)— модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. ER-модель — це мета – модель даних, тобто засіб опису моделей даних. Існує ряд моделей для представлення знань, але одним з найзручніших інструментів уніфікованого представлення даних, незалежного від програмного забезпечення що його реалізує, є модель «сутність-зв'язок». Важливим є той факт, що з моделі «сутність-зв'язок» можуть бути породжені всі існуючі моделі даних (ієрархічна, мережева, реляційна, об'єктна), тому вона є найзагальнішою.[2]

Після моделювання предметної області важливим етапом є моделювання та розроблення логічної моделі даних, за допомоги Erwin. За допомоги розробки логічної моделі даних ми зможемо висвітлювати основну сутність, зв’язки та атрибути проекту. Візуалізуючи систему, розуміючи всі аспекти та нюанси розробки, ми можемо розробити ER-діаграму, що в майбутньому дасть нам змогу рухатись далі в розробці проекту.

Логічна модель даних дасть змогу користувача розуміти специфіку роботи та структуру, що доповнюють його елементи, а саме основні елементи що маються:

Логічна модель даних має свої елемент, завдяки, якім користувач, що не розуміє структуру, йому дасть поняття такі елементи, як:

1) Сутність – будь-який помітний об’єкт (об’єкт, який можна відрізнити від іншого), інформацію про який необхідно зберігати в базі даних. Сутністю можуть бути люди, місця, літаки, рейси, смак, колір і т.д. Необхідно розрізняти такі поняття, як тип сутності та екземпляр сутності. Тип сутності

відноситься до набору однорідних осіб, предметів або подій, які виступаючих як ціле. Екземпляр суті відноситься до конкретної речі в наборі. Наприклад, типом сутності може бути МІСТО, а екземпляром – Київ і т.д.

2) Атрибут – пойменована характеристика сутності. Його найменування повинне бути унікальним для конкретного типа сутностів, але може бути однаковим для різного типа сутності (наприклад, КОЛІР може бути визначений для багатьох суті: СОБАКА, АВТОМОБІЛЬ, ДИМ). Атрибути використовуються для визначення того, яка інформація повинна бути зібрана про сутність. Прикладами атрибутів для суті АВТОМОБІЛЬ є ТИП, МАРКА, НОМЕРНИЙ ЗНАК, КОЛІР і т.д. Тут також існує відмінність між типом і екземпляром. Тип атрибуту КОЛІР має багато екземплярів або значень: Червоний, Синій, Банановий і т.д., проте кожному екземпляру суті привласнюється тільки одне значення атрибуту.

3) Ключ – мінімальний набір атрибутів, по значеннях яких можна однозначно знайти необхідний екземпляр сутності. Мінімальність означає, що виключення з набору будь-якого атрибуту не дозволяє ідентифікувати сутність по тих, що залишилися.

4) Зв’язок – асоціювання двох або більш сутностей. Якби призначенням бази даних було тільки зберігання окремих, не зв’язаних між собою даних, то її структура могла б бути дуже простою. Проте одна з основних вимог до організації реляційної бази даних – це забезпечення можливості відшукання однієї сутності за значеннями інших, для чого необхідно встановити між ними певні зв’язки. А оскільки в реальних базах даних нерідко містяться сотні сутностей, то теоретично між ними може бути встановлено більше тисячі зв’язків. Наявність такої безлічі зв’язків і визначає складність інфологічної моделей. [3]

Для проектування логічної моделі було прийнято рішення використовувати програму ERWIN Data Modeler. Саме цей підхід використання допоможе на зрозуміти та спроектувати майбутню Базу Даних.

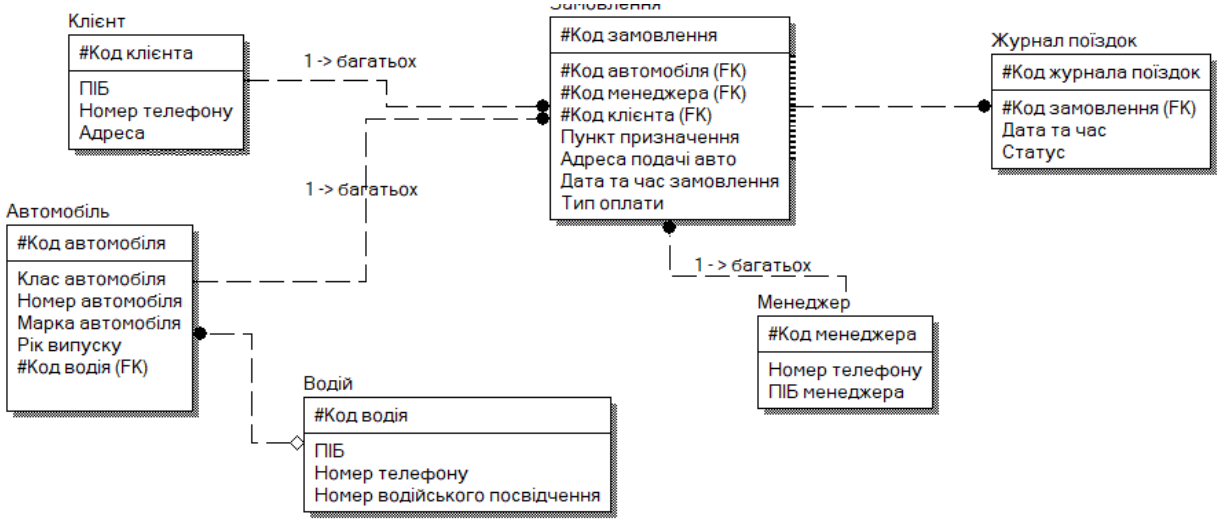

Рис. 8 Логічна модель даних

Логічна модель даних, що зображено на рис. 8, відображає 6 сутностей, нижче буде детально описано за кожну сутність та зв’язок між ними:

1) *Клієнт* – користувач системи: має ключовий атрибут #Код клієнта, що є його ідитифікатором користувача, наступне що ми можемо побачити, що клієнт має 3 неключових атрибути, а саме: ПІБ, номер телефону та адресу. Згідно з моделі, один клієнт зможе створювати багато замовлень, але кожне замовлення належить тільки одному клієнту, що є його автором. Також, аналізуючи у майбутньому, було додатково прийнято рішення додати атрибути оцінювання. Система зможе мати узагальнений середній результат оцінювання від водія. Це зроблено буде для того, щоб у майбутньому інші водії бачили рейтинг клієнта;

2) *Менеджер* – одна із ключових ролей: сутність менеджера в системі має коротке, але важливе значення. Ми бачимо ключовий атрибут ідентифікатор, та два неключові атрибути: номер менеджера для зв’язку з менеджером та ПІБ менеджера. Його роль в системі буде мати таку, як обробка замовлень, як на вхідну лінію та робота з корпоративними клієнтами;

3) *Автомобіль* – загальна підведена сутність під автомобілі: маємо ключовий атрибути ідентифікатор, та 5 неключових атрибутів. Перший із неключових атрибутів – це клас авто, номер авто, марка, рік та ідентифікатор сутності водія. А саме маємо один водій має лише один автомобіль для роботи;

4) *Журнал поїздок* – системна інформація для диспетчера. Маємо ключовий атрибут ідентифікатор, та 2 неключових атрибути та 1 ідентифікатор, який зв'язаний з замовленням. Дата та час, статус та маємо замовлення що поєднується одне замовлення до узагальнення для журналу поїздки;

5) *Замовлення* – це і є одна із найголовніших сутностей цього проекту, через цю сутність проходить майже вся система. Ми будемо мати ключовий атрибут – це ідентифікатор та 4 неключових і 3 ідентифікатора, що пов'язують інші сутності з нашою сутністю Замовлення. Із основних: пункт призначення, адреса подачі, дата та час, тип оплати. Саме 3 ідентифікатори, що пов'язують – це код автомобіля, менеджера, клієнта. Замовлення буде себе містити менеджера, який призначив, автомобіль, що призначений на це замовлення, та саме, який клієнта;

6) *Водій* – остання сутність, що буде містити в собі ключовий атрибут код водія, для відстеження та інших функцій у системі. Маємо також три неключових атрибути: ПІБ водія, номер телефону, номер водійського посвідчення.

2.2 Вибір системи управління інформаційною базою

Ключовий етап проєктування інформаційної системи є вибір системи управління базами даних (СУБД), оскільки саме вона буде забезпечувати зберігання, обробку та керування інформації, що буде використовувати наша майбутня система. Коректність вибору СУБД залежить від швидкодія системи, надійність збереження даних, масштабованість та безпечність роботи з даними.

В межах розроблювальної ІС служби диспетчера таксі база даних використовується в якості зберігання інформації та обробки. Зберігаючи інформацію ми зможемо працювати з таблицями користувачів, водіїв, Т/З, замовленнями, статусами та способами сплати за замовлення. Оскільки система передбачає роботу з декількома типами користувачів, а саме: десктопний застосунок та мобільний, на платформі .NET MAUI та Windows Forms, було прийнято рішення обрати СУБД PostgreSQL, що повинна буде забезпечувати

стабільну багатокористувацьку взаємодію та підтримувати швидко та стабільно виконувати запити.

Обрана СУБД є сучасною об’єктно-реляційною системою, яка характеризується високою продуктивністю, надійністю з підтримкою великої кількості одночасних підключень. PostgreSQL є вільно розповсюджуваною системою з відкритим програмним кодом, що дозволяє використовувати її без ліцензування та додаткових витрат.

PostgreSQL - це потужна, відкрита об’єктно-реляційна система управління базами даних (СУБД), призначена для зберігання, обробки та керування великими обсягами даних. Розроблений з акцентом на розширюваність та сумісність зі стандартом SQL, PostgreSQL підтримує як реляційні, так і деякі NoSQL-функції, такі як зберігання JSON-даних та ключ-значення. Її основне призначення - надавати безпечне та надійне рішення для зберігання даних, яке дозволяє користувачам ефективно керувати складними транзакціями та виконувати аналіз даних у режимі реального часу.

Основні плюси та мінуси:

1) Висока сумісність із SQL. PostgreSQL підтримує практично всі функції стандартного SQL, що робить його зручним для розробки складних запитів та транзакцій.
2) Надійність та відповідність ACID-стандартам. Підтримка ACID (атомарність, узгодженість, ізоляція, довговічність) робить PostgreSQL стійким та передбачуваним при обробці критичних даних.
3) Розширюваність. PostgreSQL дозволяє користувачам додавати свої функції, що корисно для специфічних завдань та налаштування під унікальні вимоги.
4) Підтримка JSON і NoSQL. Крім реляційних даних, PostgreSQL також підтримує JSON-дані, що робить його зручним для гібридних додатків, де потрібні як реляційні, так і документно-орієнтовані дані.

5) Складність настройки. PostgreSQL іноді вимагає більш складного налаштування та управління порівняно з іншими СУБД, що може вимагати досвіду в адмініструванні.
6) Ресурсозатратність. Через широкий набір можливостей і сумісності PostgreSQL може споживати більше ресурсів, ніж інші СУБД, особливо під час обробки великих обсягів даних.[4]

Взаємодія застосунків із БД у середовищі .NET може використовуватися технологія Entity Framework Core, що дозволить забезпечити об'єктно-орієнтований доступ до даних, спростити процес розробки ПЗ, використовуючи технології ORM, дозволяє зменшити кількість SQL коду, та підвищити швидкість реалізації функціональних можливостей розроблювальної системи.

Таким чином, використання PostgreSQL, у розроблювальній ІС є доцільним рішенням використання та застосування, оскільки усі розроблювальні функції задовольняють усі необхідні процеси та аспекти розробки ПЗ. Гарантуючи цілісність даних та широкі можливості інтеграції з іншими ПЗ платформи .NET.

2.3 Створення інформаційної бази

Під час розробки ІС наступним ключовим етапом є проєктування та впровадження в систему розроблювальну інформаційну базу, завдяки логічній моделі даних. База даних є центральним компонентом системи, як наприклад у автомобіля. Автомобіль не може поїхати без двигуна, так само і не ІС не можливо зробити без БД. База даних від себе забезпечує зберігання даних, обробку інформацію про користувачів, водіїв, Т/З, замовлення та платежі.

Для створення та керування БД використовується структуровані запити мовою SQL(Structured Query Language). Конкретна ця мова призначена для взаємодії з реляційними базами даних та дозволяє виконувати створення, оновлення, модифікація даних, запити та керуванням доступів користувачів.

SQL є спеціалізованою мовою роботи з БД, що забезпечує можливість конкретних операцій роботи з структурованими даними, створення об'єктів БД, наприклад: таблиці, процедури, тригери та користувачів системи.

SQL — це мова структурованих запитів у таблиці баз даних, яка забезпечує взаємодію з СУБД по всьому світу. SQL шукає інформацію навіть у big data, а також допомагає витягувати й оновлювати частини баз даних.

За роки свого існування SQL серйозно еволюціонував, як через стандарти, так і під впливом потреб практики. З роками він навчився:

- Аналізувати дані, а не лише зберігати їх — завдяки віконним функціям, CTE, підзапитам і складним агрегатам SQL став основою BI та аналітики.
- Працювати з новими форматами — підтримка JSON і XML дозволила працювати з напівструктурованими даними без необхідності переходити до NoSQL.
- Обробляти просторові дані — геометричні типи та просторові індекси відкрили SQL дорогу в геоаналітику, логістику, картографію.
- Розширюватися через розробників — більшість СУБД сьогодні дозволяють писати власні функції на популярних мовах (наприклад, Python або JavaScript), що робить SQL більш гнучким.

Типи СУБД та діалекти SQL:

SQL — це не одна система, а мова, яку підтримує безліч різних баз даних. Вони відрізняються підходом до зберігання, масштабування та аналізу даних. Розуміння цих відмінностей допомагає обрати правильний інструмент під конкретне завдання.

- Реляційні СУБД. Це класичні бази даних, які зберігають дані у вигляді таблиць із чітко визначеними зв’язками між ними. Вони ідеальні для більшості застосунків: сайтів, внутрішніх систем, CRM, фінансових сервісів тощо. Найпопулярніші рішення включають PostgreSQL, MySQL, Microsoft SQL Server, Oracle та інші.

- Аналітичні/колонкові СУБД. Такі бази оптимізовані під аналітику великих обсягів даних. Вони зберігають інформацію не рядками, а стовпцями, що робить агрегації та фільтрацію значно швидшими. Серед популярних продуктів — ClickHouse, Amazon Redshift, Google BigQuery, Snowflake.
- Гібриди, NewSQL і NoSQL з підтримкою SQL. NewSQL поєднують транзакційність класичних баз із масштабованістю NoSQL. А деякі NoSQL-системи додають SQL-подібні інтерфейси, щоби спростити роботу з даними. В цьому контексті часто користуються CockroachDB, TiDB, Google Spanner, Cassandra (CQL), MongoDB.[5]

Початковим етапом реалізації інформаційної бази є створення саме нової, чистої БД у середовищі PostgreSQL. Саме під час створення бази даних визначаються основні параметри зберігання даних та формується вже сама структура майбутньої системи. Наша майбутня база даних повинна містити та забезпечувати надійне зберігання інформації, підтримку багатокористувацького режиму роботи та швидке виконання запитів SQL.

Розпочинаючи будувати нашу майбутню базу даних, нам потрібно створити саму БД в PostgreSQL. На рис. 9 буде зображено код створення бази даних.

```
CREATE DATABASE Taxi;
```

Рис. 9 Код створення бази даних

Створення таблиці дає нам наступний крок реалізації, створення таблиць, сама база даних – це ядро, яке дає функціонуванню нашій системі. Реалізація створення таблиці має один етап та головне 2 речі, які повинно знати – це розуміння кодування за допомогою мови SQL та те що ми спроектували раніше за допомогою Erwin – це наша майбутня структура БД.

Далі буде демонстровано на рис. 10 та 11 декілька фрагментів код створення таблиць, а саме: таблиця користувачів та замовлення. Для

демонстрації в основній частині пояснювальної записки було обрано їх, тому що ці таблиці одні із найголовніших. Наступні коди створення таблиць буде продемонстровано в додатку А.

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50)
UNIQUE NOT NULL,
    password TEXT NOT NULL,
    role_id INT REFERENCES
roles(id) ON DELETE SET NULL
);
```

Рис. 10 Код створення таблиці Користувачі.

При початковій розробці було прийнято розробити саме таку модель таблиці. Де присутній ідентифікатор, що буде відзначати конкретного користувача, username – що описує користувача, а саме його ПІБ, пароль, що має шифрування в системі та таблиці та його роль в системі. Додатково, щоб розуміти, як клієнтів так і для водіїв було розроблено систему оцінювання, що надає зрозуміти загальний рейтинг водія або самого клієнта.

Наступний фрагмент коду таблиці буде замовлення. Саме ця таблиця вже є, як кінцева планка через, яку буде йти основа системи. Клієнт замовляє, сплачує, водій бачить усю необхідну інформацію, менеджер бачить статуси тощо.

```
CREATE TABLE orders (
    id SERIAL PRIMARY KEY,
    client_id INT REFERENCES
clients(id),
    manager_id INT REFERENCES
managers(id),
    driver_id INT REFERENCES
drivers(id),
    car_id INT REFERENCES
cars(id),
    pickup_address TEXT,
    destination_address TEXT,
    status VARCHAR(50),
    created_at TIMESTAMP
DEFAULT CURRENT_TIMESTAM
);
```

Рис. 11 Код створення таблиці замовлення.

Наступним етапом, потрібно розробити процедури та тригери, що дають змогу зменшити роботу в майбутньому, так, як у нас вже буде готовий інструмент для роботи в системі. На рис. 12, 13, буде зображено процедуру створення пошуку автомобілів за класом та створення тригерів що дають автоматизацію обробки подій у БД.

Тригери – особливий інструмент SQL-серверу, що використовується для підтримки цілісності даних в базі даних. За допомогою обмежень цілісності, правил і значень за умовчанням не завжди можна добитися потрібного рівня функціональності. Часто вимагається реалізувати складні алгоритми перевірки даних, що гарантують їх достовірність і реальність. Крім того, іноді необхідно відстежувати зміни значень таблиці, щоб потрібним чином змінити зв'язані дані. Тригери можна розглядати як свого роду фільтри, вступаючи в дію після виконання всіх операцій відповідно до правил, стандартних значень і т.д.[6]

Системні збережені процедури призначені для виконання різних адміністративних дій. Практично всі дії по адмініструванню серверу виконуються з їх допомогою. Можна сказати, що системні збережені процедури є інтерфейсом, що забезпечує роботу з системними таблицями, яка, кінець кінцем, зводиться до зміни, додавання, видалення і вибірки даних з системних

таблиць як призначених для користувача, так і системних баз даних. Системні збережені процедури мають префікс sp_, зберігаються в системній базі даних і можуть бути викликані в контексті будь-якої іншої бази даних.[7]

```
        CREATE OR REPLACE
PROCEDURE show_cars_by_class(
            IN selected_class VARCHAR
        )
        LANGUAGE SQL
        AS $$
            SELECT *
            FROM Cars
            WHERE car_class =
selected_class
            AND status = 'Вільний';
        $$;
```

Рис. 12 Код створення процедури пошуку автомобіля за класом.

```
        CREATE OR REPLACE FUNCTION
notify_dispatcher_about_order()
        RETURNS TRIGGER AS $$
        BEGIN
            INSERT INTO Notifications(message,
created_at)
            VALUES (
                'Створено нове замовлення таксі',
                CURRENT_TIMESTAMP
            );

            RETURN NEW;
        END;
        $$ LANGUAGE plpgsql;
```

Рис. 13 Код створення функції тригеру.

3 Прикладне програмне забезпечення

3.1 Організаційна структура програмного забезпечення

Наступним чином, наступає розробка діаграму пакетів, що буде відображати архітектуру нашого проекту. Етап розробки з організації структури ПЗ потребує розподілити програмне рішення на компоненти. Це є необхідною частиною, що між командою або учасниками процесу, для повторного використання та подальшого використання коду.

Засобом моделювання в UML, який дає змогу агрегувати окремі компоненти в групи залежно від їх логічного призначення в системі є поняття пакету (package).

Кожен пакет володіє усіма елементами, що належать до нього, при цьому сам пакет може входити до складу іншого пакету. Графічно пакет позначається відповідним прямокутником, який має унікальне ім'я та може містити в собі інші пакети. [8]

Розроблювальна діаграма пакетів представляє собою реалізацію трьохрівневу архітектуру, що дозволяє чітко розмежувати інтерфейс користувача, розробити бізнес – логіку та рівні збереження даних. На рис. 14 буде зображено діаграму пакетів розроблювальної інформаційної системи обліку замовлення таксі.

В основі рішення покладено принцип модульності, де клієнтська частина представлена у вигляді WinForms – додатка та мобільного застосунку .Net MAUI. Завдяки цьому, це дозволяє мені забезпечити швидку реакцію інтерфейсу та зручну взаємодію користувача з системою без прямих доступів до БД, підвищуючи безпеку рішень.

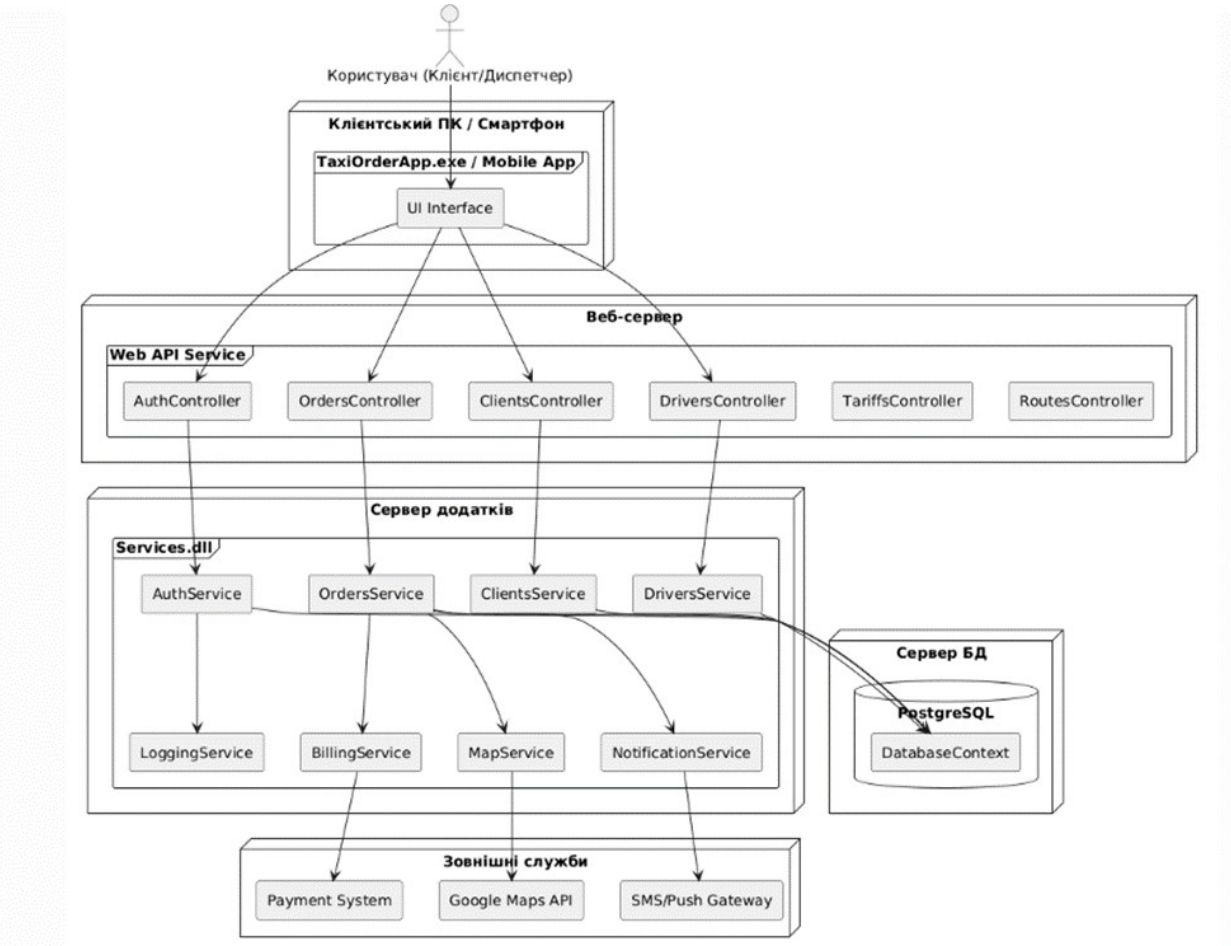

Рис. 14 Розроблювальна діаграма пакетів системи.

На рівні Веб-сервера, проєктування набору контролерів (Web API), що виконують роль проміжного шару, кожний контролер має відповідати за свій процес, а саме функціональний блок: робота із замовленнями, робота з управлінням або ж аутентифікацію. Саме такий підхід був реалізований, щоб масштабувати, а насамперед додавання нових функцій, що не дозволить порушити роботу вже налаштованих модулів.

Основна увага була приділена роботі з сервер додатків, саме тут основана вся бізнес-логіка процесів. Реалізовано окремі розділи сервісів по ключовим операціям: розрахунок по вартості проїзду, підбор водіїв та побудова маршрутів.

Головним процесом роботи для водія – є актуальна передача геоданих. В системі передбачено взаємодію з картографічними API для побудови коректного маршруту. Додатково було додано модулі з роботою платіжними шлюзами, та

сервісами сповіщення, що дає автоматизацію процесу оплати та інформування клієнта.

Взаємодія з БД відбувається через спеціальний контекст даних, що дає нам змогу гарантувати цілісність інформаційних даних, що є безпосередньо головним чинником, а навіть фактором роботи в цій сфері.

Останок процесу, впроваджено систему логування, що супроводжує всі ключові процеси. Перш за все, зроблено систему логування функціонуватиме для відстеження життєвого циклу кожного замовлення для оперативних виявлені можливих помилок. Комплексний підхід дозволяє стабільно розробити ІС та підготувати до реальних навантажень та перевіряти роботу в умовах реального часу.

3.2 Вибір інструментарію для створення прикладного програмного забезпечення

Людина рухається на крок перед, тому сучасний крок дає нам змогу приймати сучасні рішення. Для реалізації ІС обліку замовлення таксі було обрано сучасні методи та програми, комплексний підхід, що дозволяє забезпечувати стабільність, зручність розробки та можливість для подальшого масштабування продукту. Особливість враховувалась для кожного учасника процесу, мобільний застосунок для клієнтів та водіїв, а для диспетчера – десктопний застосунок.

Створення мобільного застосунку використовувало платформу .NET MAUI. Саме ця технологія була обрана мною тому, що платформа є коректною для розробки ІС системи. Один із факторів, розробка кросплатформених застосунків з єдиною кодовою базою, що значно дозволяє спростити процес розробок та підтримок ПЗ. Використання .NET MAUI забезпечує можливість запуску застосунку на різних платформах, а саме головне, сучасний та адаптивний інтерфейс користувача.

Основні переваги використання платформи .NET MAUI у межах розроблювальної системи є:

1) Можливість створення єдиної бази для різних платформ, IOS, Android, що дуже тішить при розробці мобільного застосунку;

2) Підтримка сучасних технологій інструментів платформи та інтеграція з іншими сервісами Microsoft;

3) Реалізація графічного інтерфейсу, який дає змогу розробити його сучасним для користувача, за допомоги XAML.

4) Підтримка роботи з API, з базами даних та мережевими сервісами, що є великою складовою системи;

5) На останок, перевагою складає продуктивність та мобільність створення адаптивного інтерфейсу для мобільних пристроїв.

Наступним кроком є реалізація роботи диспетчера, що буде використано технологію розробки Windows Forms. Використання десктопного застосунку для диспетчера також є невід'ємною частиною складової розроблювальної системи, а саме централізована обробка замовлень та забезпечення ефективним керування процесу перевезень.

Переваги Windows Forms у межах проекту є:

1) Простота створення графічного інтерфейсу користувача з швидкою реалізацією функціональних елементів системи;

2) Велика кількість вбудованих компонентів для роботи з таблицями, кнопками тощо;

3) Зручна інтеграція з платформою . NET MAUI та БД;

4) Велика зручність в швидкому тестуванні та налагодження системи;

5) Підтримка подій та обробників, що дозволяє реалізувати логіку взаємодії з диспетчером та замовленням разом з водієм.

Загально, для розробки проєкту було обрано реалізацію ПЗ з мовою C#. Саме ця мова є однією з найбільш поширених технологій для створення застосунків на платформі .NET, і насамперед забезпечити підтримку ООП підходу до розробки.

3.3 Алгоритмізація та програмування програмних модулів

Алгоритмізація та програмування програмних модулів є загальна частина розробки, а саме демонстрація розроблення процесу у вигляді блок-схеми. Для подальшої розробки потрібно продумати алгоритми дії певних користувачів системи, щоб зрозуміти правильність логікі дії системи та дій користувача. Саме від цього залежить, правильність функціонування процесу.

Саме алгоритм дозволяє бачити певні помилки виконання дій, вирішувати їх на місці та подальшому впровадити у систему. На рисунку 15 та 16 буде реалізовано блок-схему обробки даних та формування звітно-статистичної інформації.

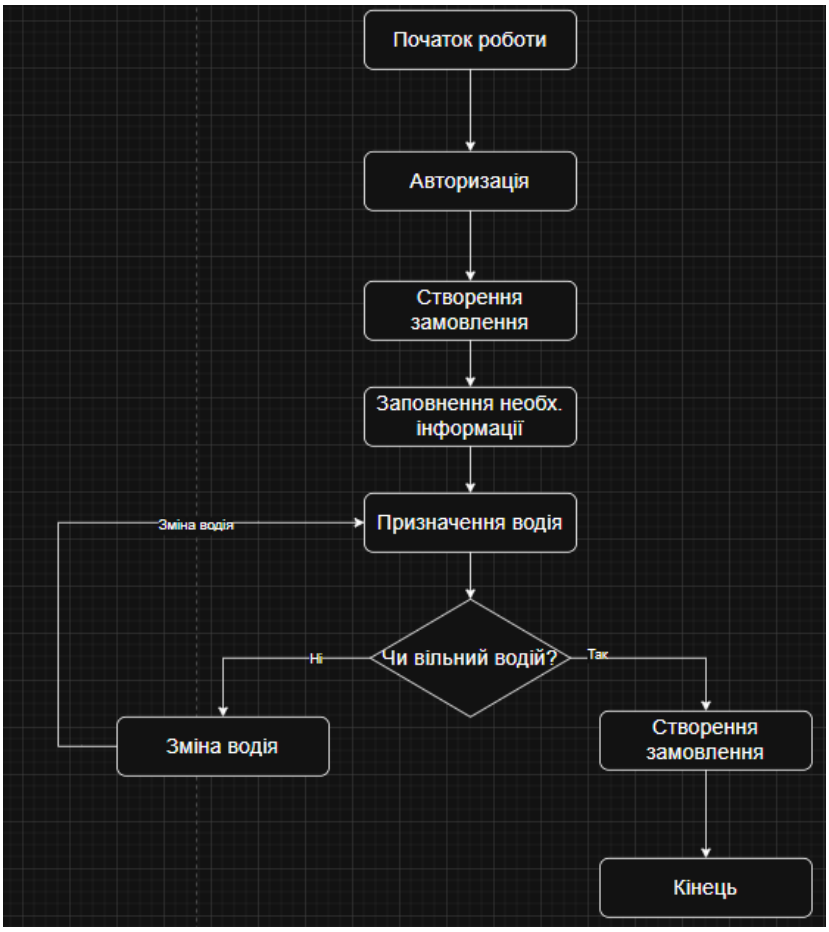

Рис. 15 Алгоритм обробки даних.

Саме тут реалізовано процес дій формування замовлення, що дає обробку даних. Ми можемо побачити початок та кінець дій, розпочинаючи з обов’язкової авторизації, та наступним процесом створення замовлення з заповненням необхідних даних для створення. Наступним фактором вступає призначення

водія де і діє перевірка, чи вільний водій, якщо ні, його не буде продемонстровано диспетчеру, якщо водій вільний то ми приступаємо до створення замовленні та завершуємо алгоритм.

Завдяки цьому алгоритму було реалізовано створення замовлення з обробкою даних в системі.

Наступним реалізація блок-схеми формування звітно-статичної інформації, що буде зображено на рис. 16. Також зможемо побачити обов'язкову авторизацію у систему, де ми переходимо до категорії звітів та переглядаємо конкретний звіт, наприклад звітність по поїздкам, що продемонстровано на рисунку. Перевірка дії йде на коректність дати за відбором, якщо дата актуальна переходимо до демонстрації актуальності звітності по поїздкам та завершуємо процес.

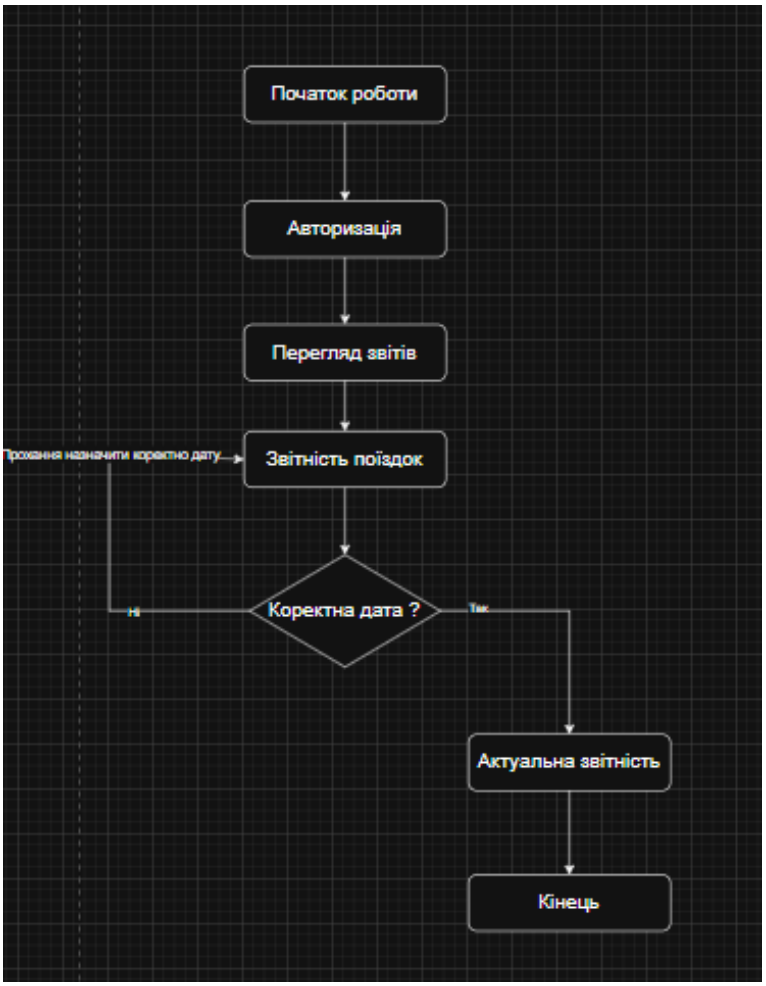

Рис. 16 Алгоритм формування звітно-статичної інформації.

За допомогою цих блок-схем було впроваджено та реалізовано в десктопному застосунку, що буде зображено на рис. 17 та 18 саме у вигляді кодової частини. Реалізаціє не є складною, але і не простою, саме від цього подальшому залежить процес роботи.

```
using (var conn = new NpgsqlConnection(connString))
{
    conn.Open();

    string sql = @"
        INSERT INTO orders
        (client_id, driver_id, car_id, pickup_address,
destination_address, status)
        VALUES
        (@client, @driver, @car, @pickup, @dest, 'Active')";

    using (var cmd = new NpgsqlCommand(sql, conn))
    {
        cmd.Parameters.AddWithValue("@client",
selectedClientId);
        cmd.Parameters.AddWithValue("@driver", driver.Id);
        cmd.Parameters.AddWithValue("@car", car.Id);
        cmd.Parameters.AddWithValue("@pickup",
txtPickup.Text);
        cmd.Parameters.AddWithValue("@dest",
txtDestination.Text);
```

Рис. 17 Фрагмент коду створення замовлення.

```
string sql = @"
    SELECT status, COUNT(*)
    FROM orders
    WHERE created_at BETWEEN @from AND @to
    GROUP BY status";

using (var cmd = new NpgsqlCommand(sql, conn))
{
    cmd.Parameters.AddWithValue("@from", from);
    cmd.Parameters.AddWithValue("@to", to);

    using (var reader = cmd.ExecuteReader())
    {
        while (reader.Read())
        {
            string status = reader.GetString(0);
            int count = reader.GetInt32(1);

            if (status == "Done")
                done = count;
            else if (status == "Cancelled")
                cancelled = count;
```

Рис. 18 Фрагмент коду звітності.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Наступний етап – тестування системи. Перед публікацією та використанням майбутніх користувачів, система потребує тестування, що є обов'язковим пунктом розробки. Завдяки тестуванню можливо виявити ті помилки, що не змогли проявитись під час розробки або були упущенні моменти. Важливо розуміти, що під час тестування, необхідно перевіряти за пунктами: цілями та обсягом, що чітко визначено та тестується, а що виходить за його рамки, далі стратегія, ресурси та інструменти, критерії входу та виходу(наприклад можлива кількість багів у системі), ризики та звітність.

Навантаження системи, чітко дозволяє розуміти великий обсяг, що може давати необмежена кількість користувачів на систему. Треба розуміти усі аспекти навантаження та давати безперебійну роботу системи від цього залежить чи коректно буде працювати диспетчер, водій або основний клієнт системи. Ризик втратити прибутку, так, як ціль кожного продукту заробити.

Проведення підсумків тестування ми підводимо до майже кінцевого результату. Швидке вирішування проблем, усунення за допомогою усіх допоміжних компонентів та на кінець перевірити функціональність та коректність роботи системи.

Особлива увага повинна приділятись основним задача, що задавались під час проєктування системи. Якщо не виконувати основні задачі тоді проєкт не буде виконаний за стандартами, цілями та метою.

Наступним етапом – є перевірка навантаженої системи, що дає розуміння та проявлення ситуації при завантаженості системи, як себе буде проявляти функціональність, та коректність роботи.

Для тестування проекту також було залучено етап регресійного тестування, даючи на змогу оглянути з усіх боків, надати поточні проблеми системи та усунути ці недоліки. Під час тестування можливо з'являться наступні недоліки під час усунення основних недоліків. Наперед, як із системою

оцінювання. Розробка повинна реалізовувати для двох користувачів системи, а саме клієнт та водій. Але питання постає в тому, що клієнт та водій бачать загальний рейтинг, оцінюють один одного після поїздок, але обов'язково є те, що клієнт має бачити загальний рейтинг водія під час поїздки. Саме це було виявлено під час тестування.

Чому саме потрібно тестувати ? Тестувати потрібно не лише для прояву помилок у системі, а і для доповнення її саме системи. Що для мене є необхідною частиною. Кожен продукт не є ідеальним. Не можливо довести продукт до самовдосконалення або ж мати його ідеальним з точки зору клієнта. Тому і розробники планують, масштабують та тестують.

Під час розробки було багато чого доповнено до системи. До прикладу буде наведено створення сплати банківською картою. Під час розробки можливо розробити двома способами за допомогою коду розробити фіктивну форму сплати у системі за допомогою випадковості проставити статуси сплати «Недостатньо коштів» або «Сплачено». Моє вирішення цього питання було таким, реалізувати приближений до реальної форми сплати за допомоги українського сервісу онлайн сплати LiqPay. Чому саме цей сервіс був обраний для форми сплати ? Тому, що, якщо впроваджувати у реальну форму сплати та працювати на цій основі, тоді цей сервіс у майбутньому зі своєю комісією та сплатою за сервіс підійде до використання.

Наступним прикладом під час тестування було виявлено недолік, який принципово повинно доповнити до сучасності, а саме реалізація трансляції місцезнаходження водія, як до початку поїздки, на початку та кінці поїздки. Цей великий плюс надає сучасності нашій системі.

Завершаючи та ключовий етап розробки перевірка на сумісність, що входить до рядів тестування, та є обов'язковим компонентом участі в цьому етапі. Наша реалізація пропонує бути сумісним одночасно для диспетчера за допомогою десктопного застосунку, реалізуючи йому автооновлення за допомогою місця на GoogleDrive, та мобільним застосунком, що є реалізацією,

як під ОС IOS та Android. Як описувалось в розділі 3, пункту 3.1, задумка реалізації пропонується мобільність під різні ОС.

Прикладом для пояснювальної записки було обрано також розробити юніт-тест, що на мою думку є необхідною частиною тестування. Зображено на рис. 19 юніт-тест розрахунку вартості за проїзд.

Мета та реалізації юніт-тесту перевірка коректності логіки, а саме відстань що множиться на заданий тариф диспетчером. Нижче буде описано саме цей юніт-тест по функціональності за кодом:

1) *Calculate_EconomClass_10km_Returns330* – мета перевірки, перевірити метод *Calculate* класу *TripPriceCalculator,* що повинно повертати коректну вартість за передану відстань та класу автомобіля *distanceKm = 10.0* та *carClass = «Економ»* - вхідні дані розрахунку *_calculator.Calculate(distanceKm, carClass)* виконує розрахунок вартості поїздки множачи відстань на тариф класу Економ, а саме 33 гривні за один кілометр з урахуванням мінімальної вартості за поїздку – 50 гривень та кінцевий вивід результату *Assert.That(result, Is.EqualTo(330.0))* очікуючи, що вартість поїздки 10 кілометрів класом Економ дорівнюватиме 330 гривні.

```
public class TripPriceCalculatorTest
   {
      private TripPriceCalculator _calculator;
      public void SetUp()
      {
         _calculator = new TripPriceCalculator();
      }
      public void Calculate_EconomClass_10km_Returns330()
      {
         double distanceKm = 10.0;
         string carClass   = "Економ";
         double result = _calculator.Calculate(distanceKm,
carClass);
         Assert.That(result, Is.EqualTo(330.0),
            "10 км × 33 грн/км має дорівнювати 330 грн");
      }
   }
   public class TripPriceCalculator
   {
      private readonly Dictionary<string, (double PricePerKm,
double MinPrice)> _tariffs
         = new()
         {
            { "Економ",  (33.0, 50.0)  },
            { "Комфорт", (45.0, 70.0)  },
            { "Бізнес",  (65.0, 100.0) }
         };
      public double Calculate(double distanceKm, string carClass)
      {
         if (!_tariffs.TryGetValue(carClass, out var tariff))
            throw new ArgumentException($"Невідомий клас:
{carClass}");
         return Math.Max(distanceKm * tariff.PricePerKm,
tariff.MinPrice);
      }
   }
}
```

Рис. 19 Розрахунок вартості поїздки таксі.

Завершальний етап тестування, вже демонстрацій кінцевого результату. Кінцевий результат міститиме демонстрацію та опису, як мобільного застосунку та десктопного застосунку диспетчера таксі. Починаючи з рисунку 20 та завершуючи по рисунок 32, буде продемонстровано ключові роботи над проєктування застосунку.

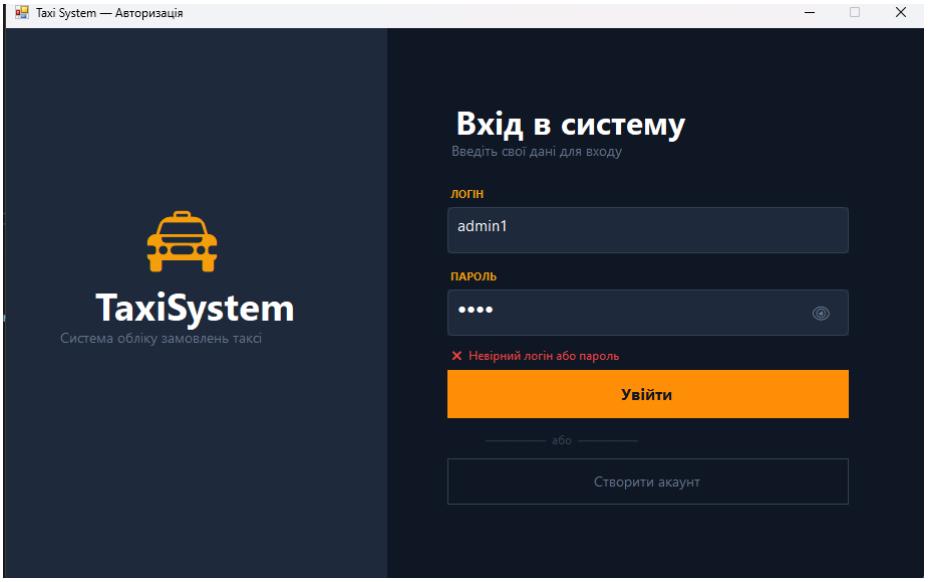

Рис. 20 Вікно авторизації десктопного застосунку з перевіркою на коректність даних

Паралельно буде демонстрація мобільного застосунку для того, щоб розуміти усю візуальність проекту та його тестувань. Саме такий підхід візуалізації тестування буде коректним для демонстрація проекту та його функцій з його повним описом функціоналом.

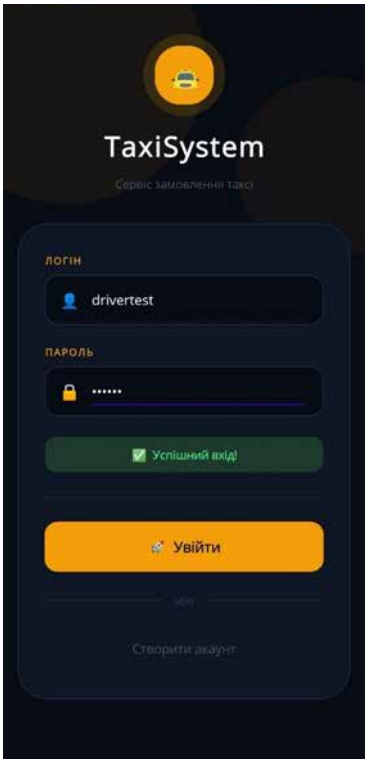

Рис. 21 Вікно авторизації із мобільного застосунку.

На одному із головних напрямків розробки – є безпека та конфіденційність даних, на рис. 20 та 21 зображено, що пароль користувачів замальований, для того, щоб інші користувачі не зрозуміли, який саме пароль вводиться.

Наступною буде демонстрація головних сторінок диспетчера, водія та клієнта. Приємно здивує те що реалізовано багато функцій, як для клієнта, водія та диспетчера. Наприклад у диспетчера відображається реальне переміщення водіїв, що дає змогу зрозуміти одразу, що водії на роботі, що вони працюють. Рисунки 22, 23, 24 відображає головну сторінку: диспетчера, водія, клієнта.

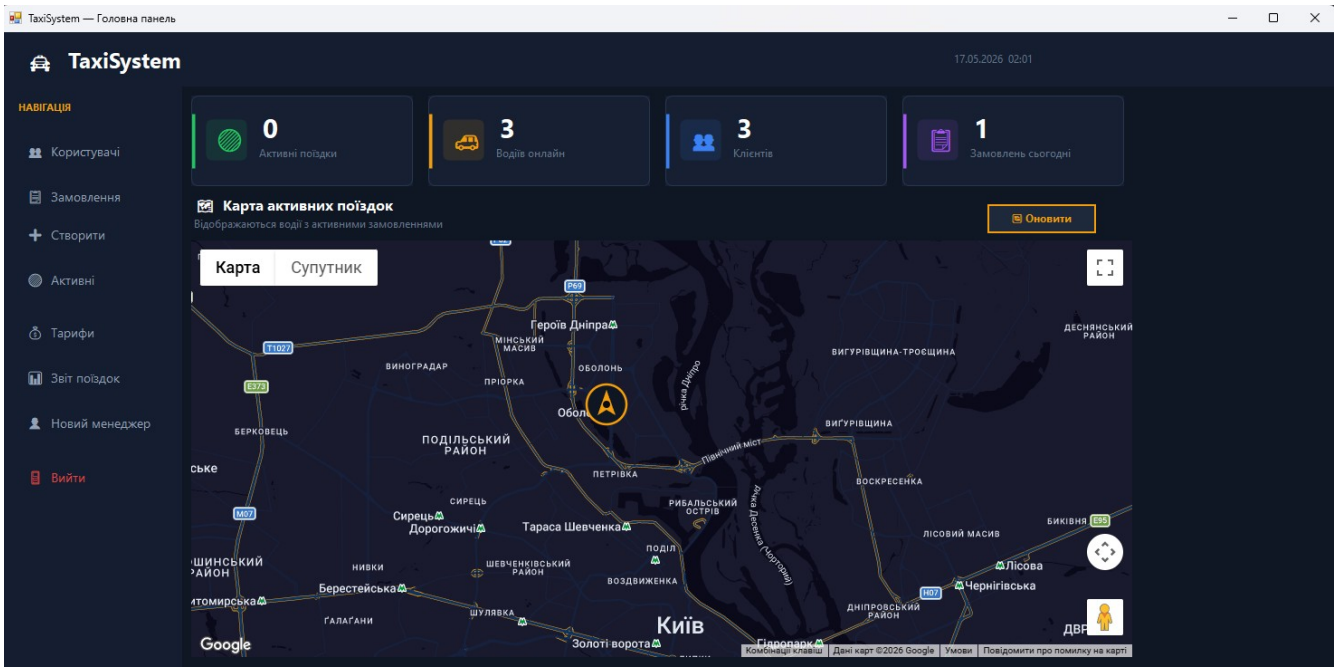

Рис. 22 Головна сторінка диспетчера.

На головній сторінці диспетчера ми побачимо статистику водіїв онлайн, активні поїздки, кількість клієнтів та замовлень за сьогодні. Бокове меню для зручності в навігації системи, а саме таблиці користувачі, замовлення, активні поїздки, створення замовлення, зміни тарифів, звіт поїздок та створення користувачів у систему. Також головна сторінка складається з відображення місцезнаходження у реальному часі водіїв. Наступна демонстрація на рис. 23 та 24.

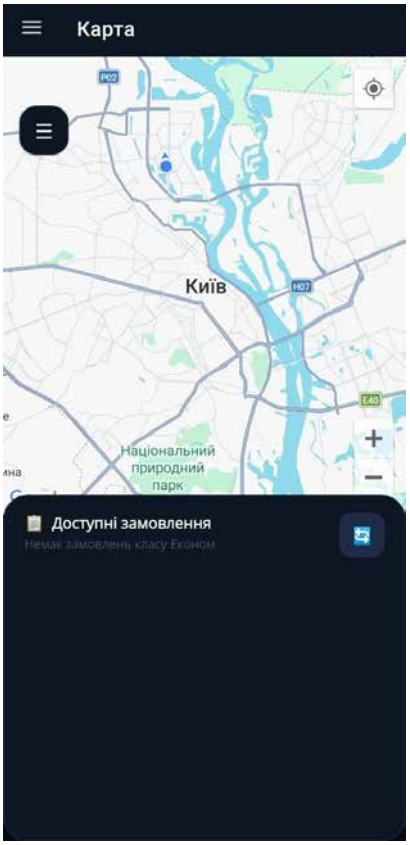

Рис. 22 Головна сторінка водія.

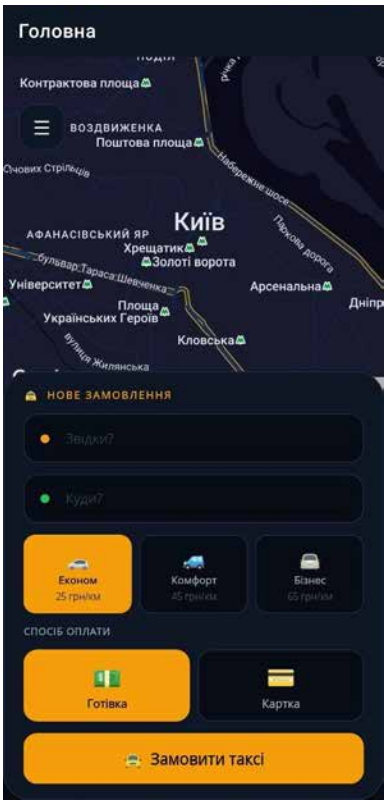

Рис. 23 Головна сторінка клієнта.

Аналітична звітність – є певним кроком для аналізу використання системи. На рисунку 24 буде зображено аналітичну статистику від менеджера з

десктопного застосунку. Аналітика та звіти містять в собі, як у рахунковому вигляді так і у звітах. Має звіти за: статусами замовлень, доходами по дням, топ водіїв та класів авто. Звітність можна проводити за періодом обраний менеджером.

Рис. 24 Демонстрація Аналітики та звітів.

Також доповнюючи аналітику та звітність для диспетчера, клієнт та водій також зможуть бачити особисту звітність та статистику в особистих кабінетах.

Наступним етапом демонстрації тестування – є сплата проїзду, що реалізована за допомогою сервісу LiqPay, як описувалось на початку пункту. На рис. 25 та 26 буде зображено процес сплати за картою.

Рис. 25 Форма оплати замовлення.

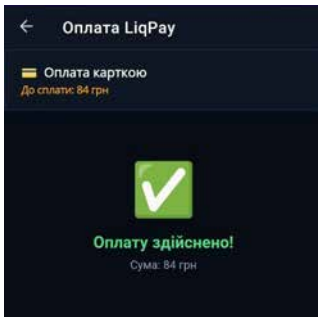

Рис. 26 Статус сплати замовлення.

Розглянемо більш детально інтерфейс зі сторони водія. Для водія на сьогоднішній день розроблено не великий функціонал, але найголовніше впроваджено туди. За головною сторінкою ми можемо побачити на рис. 22, бокове меню, що стає зручним у використані. Бокове меню складає з таких складових, як: відображення особистого загального рейтингу, карту з відображенням активних замовлень, історію виконання замовлень та профіль зі статистикою. Якщо розповідати по кожному окремо, історія виконання складає з себе перегляд як виконаних та скасованих замовлень, адреси приїзду від початку до кінця, клієнт, та заробіток який зміг заробити водій. Профіль та статистика складає з відображення власного авто, який заповняв водій після реєстрації та загальну статистику з поїздок, заробіток і рейтинг водія. Водій також може змінювати дані. На рисунку 27, 28, 29 буде зображено вище описаний функціонал.

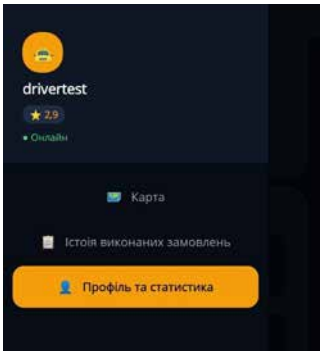

Рис. 27 Бокове меню водія.

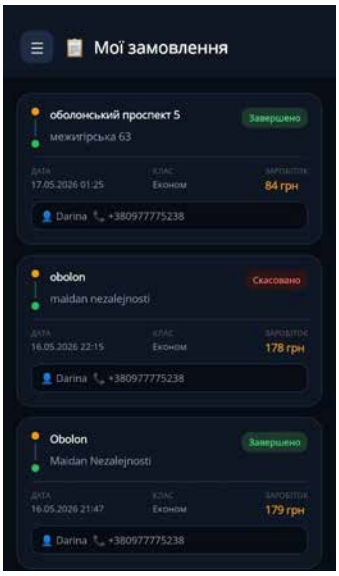

Рис. 28 Вкладка з інформацією про замовлення водія.

Рис. 29 Особиста сторінка водія із даними водія.

На останок розглянемо інтерфейс клієнта. Клієнт – це ключовий продукт системи. Саме клієнт робить замовлення, платить за нього. Цікавість та зручність проекту залежить саме розробка для нього. Клієнт оцінює як частину бекенду так і фронтенду. На рисунку 23 ми могли побачити головну сторінку клієнта, що складається з вибору сплати, типу класу авто та з точкою подачі із місцем висадки пасажира. Для зручності було розроблено бокове меню, що складається з вкладок: замовлення, історія поїздок та профілю користувача. На рис. 30, 31, 32 буде зображено функціональність для клієнта.

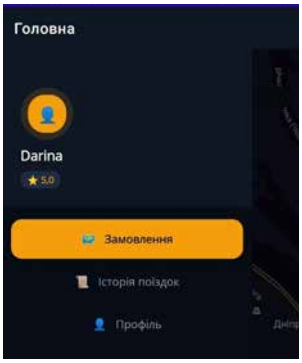

Рис. 30 Бокове меню клієнта.

Рис. 31 Історія поїздок.

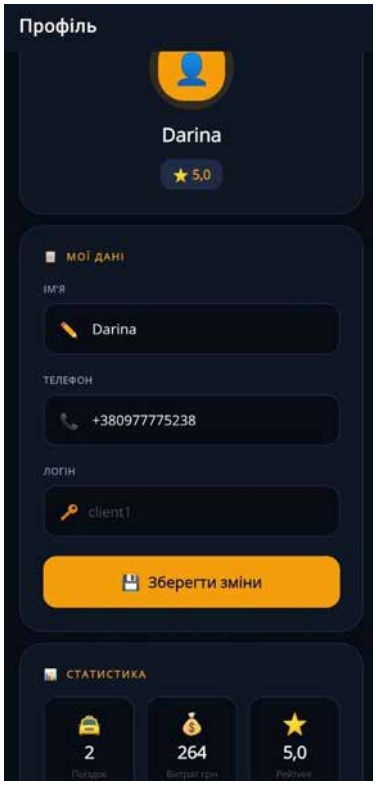

Рис. 32 Головна сторінка клієнта.

Також додатково буде продемонстровано відображення демонстрацією буде відображення перегляду користувачів системи та керування тарифами у системі для майбутнього розрахунку вартості проїзду, що це все буде зображено на рисунку 33 та 34.

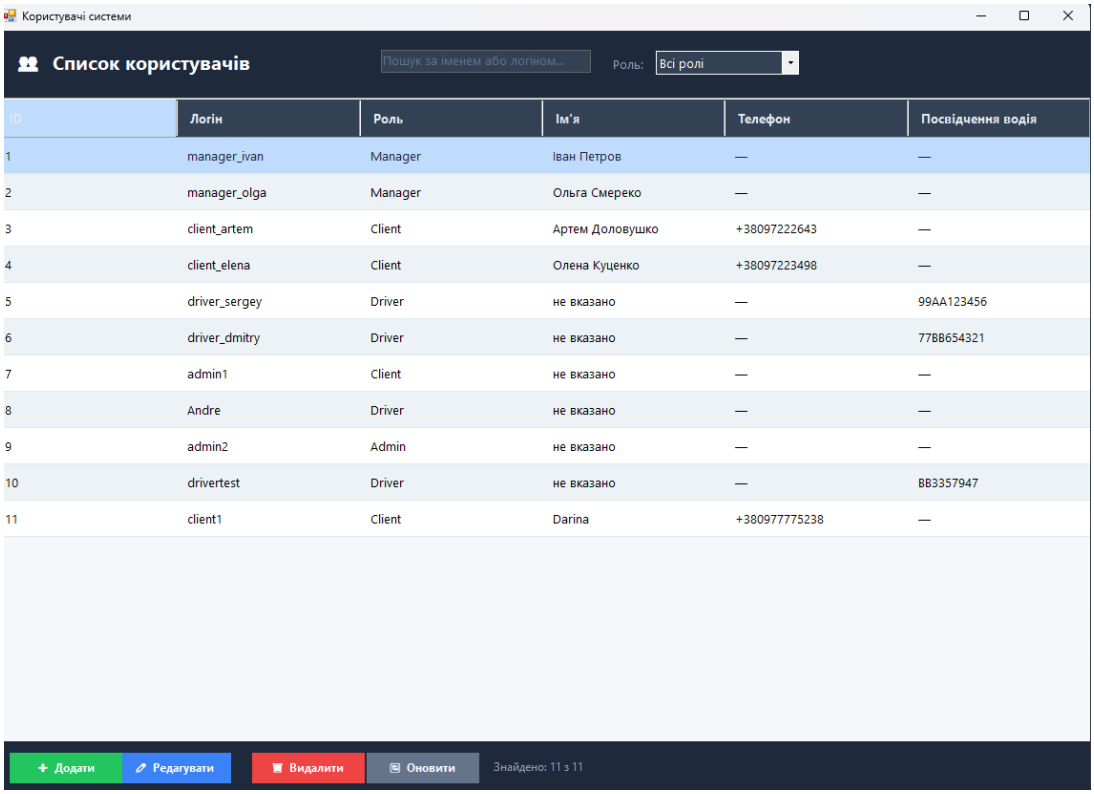

Рис. 33 Демонстрація перегляду сторінки користувачів системи

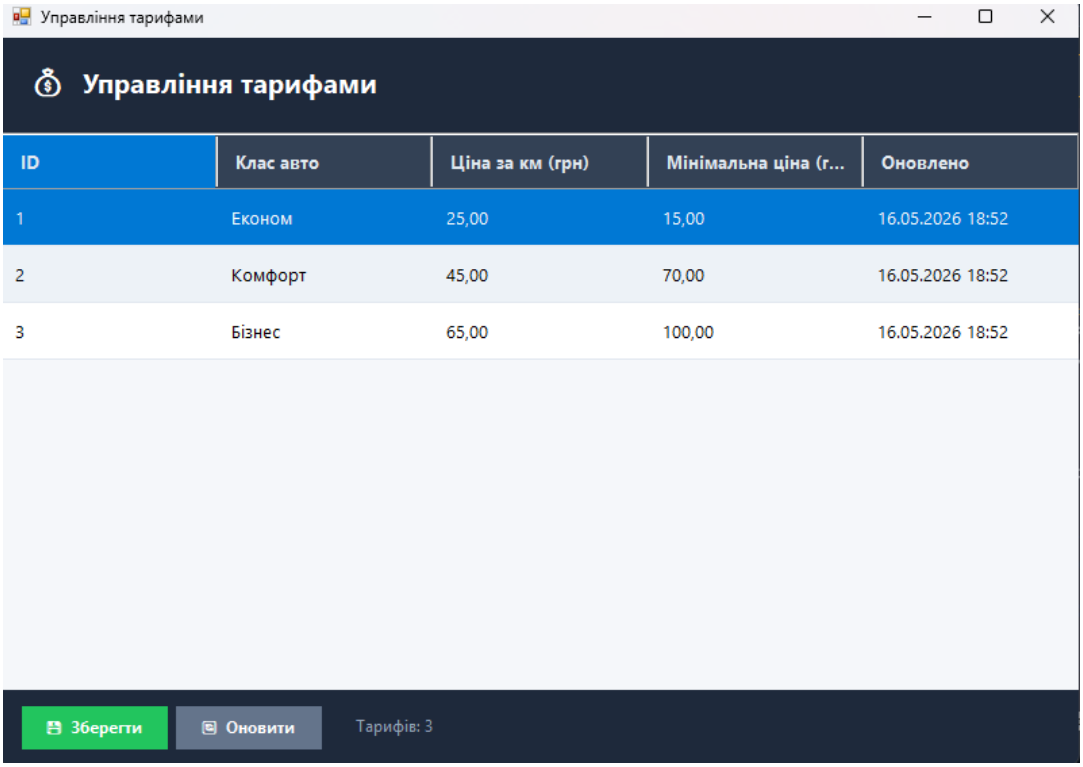

Рис. 34 Демонстрація пунктом керування зміни тарифів за проїзд.

Та останньою демонстрацією буде реалізація таблиці керування замовленнями, що буде зображено на рис. 35. Саме такою реалізацією в майбутньому буде користуватись диспетчер систем для того, що розуміти статуси активних або скасованих замовлень, навіть тих, що в дорозі.

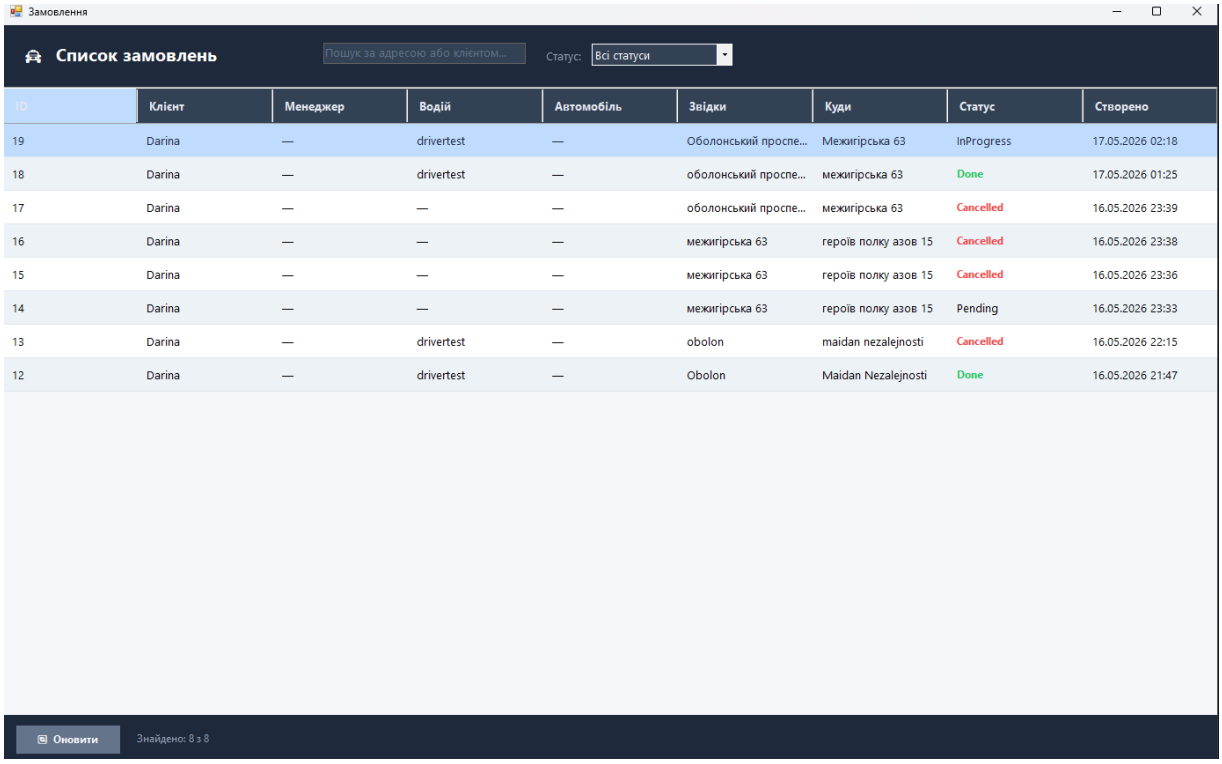

Рис. 35 Пункт керування замовленнями.

Загальність тестування та демонстрацію спроектованої системи буде представлено на захисті дипломного проекту. Де буде повноцінно зображено роботу системи десктопного застосунку та мобільного.

4.2 Вимоги до апаратного та програмного забезпечення

Для коректної безперебійної роботи ІС системи обліку замовлень таксі необхідно дотримуватись визначених вимог до апаратного програмного забезпечення. Враховуємо те, що система складається декількох системних розробок, а саме десктопного застосунку, що буде використовуватись для диспетчера або навіть можна назвати менеджера системи та мобільного застосунку, що розроблено конкретно під водія та клієнта.

Мінімальні технічні вимоги дозволяють забезпечити повноцінну роботу основних функцій системи, таких, як створення, обробки, обміну даних з сервером та найголовніше підтримка багатокористувацького режиму роботи.

До основних апаратних вимог будуть належати наступні технічні застосування:

1) Процесор – Intel Core i3 двоядерний архітектури або AMD Ryzen 3, саме процесор повинен забезпечувати достатню обчислювальну потужність для

обробки запитів SQL. Роботу клієнтського застосунку та взаємодію із сервером, що буде забезпечувати стабільну роботу системи із тактовою частотою не нижче 2.0 ГГц;

2) Оперативна пам’ять – обсяг пам’яті для розроблювальної системи буде прописаний мінімальний з урахування необхідних для одночасної роботи ОС, середовище виконання .NET, десктопного застосунку диспетчера та СУБД. Рекомендується використання 8 ГБ оперативної пам’ять або більше;

3) Місце на диску – система має використовувати не менше 500 ГБ вільного місця або навіть більше для майбутніх оновлень програм та мінімально 1 ГБ для бази даних;

4) Монітор – Використання розширення залежить також критично, для розроблювальної системи було продумано та прийнято рішення використовувати не менше з підтримкою роздільної здатності 1366x768 піскелів. Для зручної роботи рекомендується монітор з форматом Full HD;

5) Мережа – мінімально можна використовувати доступ до локальної мережі з підключенням до Інтернету.

Програмні вимоги для системи:

1) ОС – Windows 10 або більш новіша версія;
2) Платформа .NET – встановлена версія .NET 8 або новіше;
3) СУБД – PostgreSQL – версія 14 або новіша;

4.3 Склад інсталяційного пакету

Інсталяційний пакет складає з себе розуміння та представлення встановлення проєктуючої системи. Саме для цього було створено діаграму розгортання, що буде відображати ієрархію складової системи, відображення проєктування та, що від чого буде залежить в майбньому у використані.

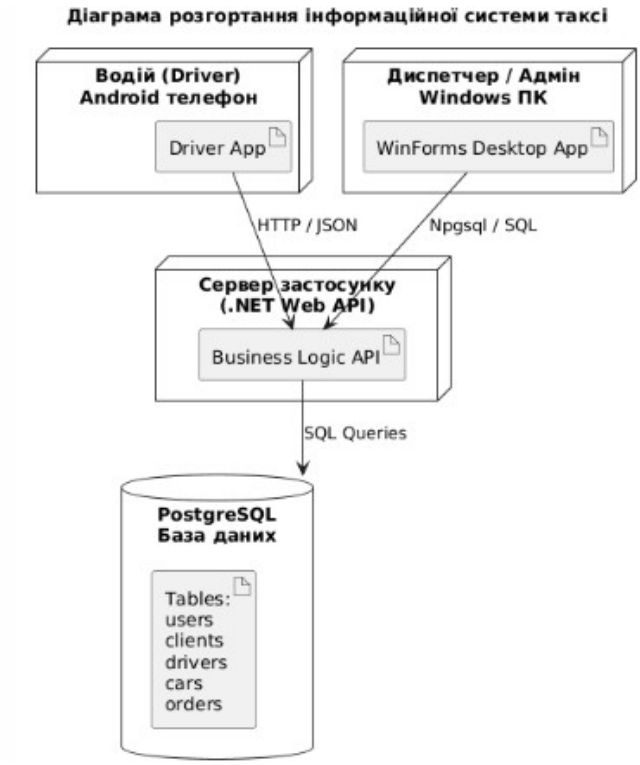

Рис. 36 Діаграма розгортання обліку замовлення таксі.

На рис. 36 відображає діаграму розгортання обліку замовлення, що має в собі БД, сервер та основа застосунки, як для водія з клієнтом так і для диспетчера. Основна бізнес – логіка закладена в сервері застосунку, що звісно прописана в самих застосунках, що буде звертатись до БД. Основа відображена саме в цьому, нічого лишнього та все зрозуміло. Ми маємо БД, до якої поступають основні запроси, далі від сервера застосунку, завдяки прописаній бізнес-логіки ми можемо оброблювати їх коректно та надавати вже далі усю необхідну інформацію, що потребувалась від диспетчера, водія або менеджера.

ВИСНОВКИ

У процесі виконання кваліфікаційної бакалаврської роботи на тему «Інформаційна система обліку замовлень послуг таксі» було здійснено перш за все дослідження предметної області у сфері пасажирських перевезень, користування транспортними послугами та узагальнити для майбутньої системи рішення розробки. В межах дослідження було поставлено завдання, визначити функціональні можливості системи, перелік конкретних процесі для потреби автоматизації, зокрема авторизацію із реєстрацією користувачів у систему,

оформлення та проведення обліку замовлення, управління інформацією, а також формування звітності.

Під час виконання роботи здійснено огляд інформаційних джерел та проведено аналіз існуючих рішень, у сфері автоматизації послуг таксі. На основі цього, нам відкриваються нові двері розробки проекту. А насамперед потрібно провести аналіз та визначити певні переваги, недоліки майбутньої системи, що надаватимуть обґрунтування рішення створення. Відображення структури проекту розроблялась за допомого побудови майбутньої моделі предметної області у вигляді UML – діаграм, а саме: діаграма прецедентів, активності, що звісно забезпечують наочне представлення взаємодій компонентів.

Процес розробки інформаційного забезпечення складається у створені логічної моделі даних, що необхідно описувати структуру основної бази системи. У проведені проектування та планування було прийнято рішення вибрати СУБД PostgreSQL, що на мою думку є необхідним в цьому проекту для функціональності усіх існуючих компонентів. Наступним чинником є те, що ми обрали структуру, обґрунтували вибір інструментальних засобів розробки, а саме за допомогою середовища Visual Studio з використанням мови програмування C#, що в **результаті було створено зручний інтерфейс користувача та реалізовано основний** функціонал системи, що дійсно дозволяє забезпечувати ефективну безперебійну роботу системи, як для менеджера, водія та клієнта.

Окремий та останній етап розробки приділяється питанням тестування програмного забезпечення, підводити визначення вимог до апаратного забезпечення та формуванню вимог до інсталяційного пакету для подальшої роботи з системою в майбутньому. Проведене тестування підтвердило працездатність усіх необхідних компонентів, стабільність та коректність виконання функцій розроблювального проекту.

Результатом бакалаврської кваліфікаційної роботи є повноцінна інформаційна система обліку замовлень послуг таксі, що дозволяє

автоматизувати основні процеси діяльності служби, забезпечувати ефективне управління даними та підвищення якості обслуговування користувачів системи. Розроблена система адаптована для використання у діяльності транспортних компаній і диспетчерських компаній для оптимізації процесу обробки замовлень та підвищення ефективності підприємства.

Список використаних джерел

1) Діаграма варіантів використання (use case diagram) [Електронний ресурс] – Режим доступу: https://studfile.net/preview/9828815/
2) Модель «сутність-зв‘язок» або er-модель [Електронний ресурс] – Режим доступу: https://studfile.net/preview/16869580/page:29/
3) Лекція 3. Інфологічна модель даних “Сутність-зв’язок” План [Електронний ресурс] – Режим доступу: https://studfile.net/preview/7363846/
4) PostgreSQL: введення та встановлення [Електронний ресурс] – Режим доступу: https://surl.li/yjqlss
5) SQL: Що це і чому без нього в сучасному IT ніяк [Електронний ресурс] – Режим доступу: https://surl.li/kvfhqx
6) Визначення тригера в стандарті язика sql [Електронний ресурс] – Режим доступу: https://studfile.net/preview/14501243/
7) Типи збережених процедур [Електронний ресурс] – Режим доступу: https://studfile.net/preview/14501231/page:2/
8) Діаграми послідовності (sequence diagram) [Електронний ресурс] – Режим доступу: https://studfile.net/preview/9877319/page:2/

Додаток А

SQL-КОД

Сторінок – 2

2026

```
CREATE TABLE roles (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) UNIQUE NOT NULL
); CREATE TABLE users (
    id SERIAL PRIMARY KEY,
```

```
    username VARCHAR(50) UNIQUE NOT NULL,
    password TEXT NOT NULL,
    role_id INT REFERENCES roles(id) ON DELETE SET NULL
); CREATE TABLE clients (
    id SERIAL PRIMARY KEY,
    user_id INT UNIQUE REFERENCES users(id) ON DELETE CASCADE,
    phone VARCHAR(20),
    name VARCHAR(100)
); CREATE TABLE managers (
    id SERIAL PRIMARY KEY,
    user_id INT UNIQUE REFERENCES users(id) ON DELETE CASCADE,
    name VARCHAR(100)
); CREATE TABLE drivers (
    id SERIAL PRIMARY KEY,
    user_id INT UNIQUE REFERENCES users(id) ON DELETE CASCADE,
    license_number VARCHAR(50)
); CREATE TABLE cars (
    id SERIAL PRIMARY KEY,
    model VARCHAR(50),
    number_plate VARCHAR(20) UNIQUE,
    color VARCHAR(30)
);
ALTER TABLE drivers
ADD COLUMN car_id INT UNIQUE REFERENCES cars(id);
CREATE TABLE orders (
    id SERIAL PRIMARY KEY,
    client_id INT REFERENCES clients(id),
    manager_id INT REFERENCES managers(id),
    driver_id INT REFERENCES drivers(id),
    car_id INT REFERENCES cars(id),
```

```
    pickup_address TEXT,
    destination_address TEXT,
    status VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
CREATE TABLE trip_logs (
    id SERIAL PRIMARY KEY,
    order_id INT REFERENCES orders(id) ON DELETE CASCADE,
    start_time TIMESTAMP,
    end_time TIMESTAMP,
    notes TEXT
);
CREATE TABLE order_trip (
    order_id INT REFERENCES orders(id) ON DELETE CASCADE,
    trip_id INT REFERENCES trip_logs(id) ON DELETE CASCADE,
    PRIMARY KEY (order_id, trip_id)
);
```

Додаток Б

КОД РЕАЛІЗАЦІЇ ОСНОВНИХ ФУНКЦІЙ

Сторінок – 8

2026

Реалізація функціональності авторизації мобільного додатку з перевіркою по ролі.

```
  private async void OnLoginClicked(object sender, EventArgs e)
{
string username = UsernameEntry.Text?.Trim() ?? "";
string password = PasswordEntry.Text ?? "";
```

```
if (string.IsNullOrWhiteSpace(username) || string.IsNullOrWhiteSpace(password))
{
ShowResult("⚠ Заповни всі поля!", isError: true);
return;
}
string hashedPassword = HashPassword(password);
try
{
using (var conn = new NpgsqlConnection(connString))
{
conn.Open();
string sql = @"
SELECT u.id, u.username,
CASE WHEN d.id IS NOT NULL AND d.license_number IS NOT NULL
AND d.license_number != '' AND ca.id IS NOT NULL
THEN true ELSE false END AS profile_complete,
r.name AS role,
COALESCE(ca.class, '') AS car_class
FROM users u
LEFT JOIN roles   r  ON u.role_id = r.id
LEFT JOIN drivers d  ON d.user_id = u.id
LEFT JOIN cars    ca ON ca.id = d.car_id
WHERE u.username = @u AND u.password = @p";
using (var cmd = new NpgsqlCommand(sql, conn))
{
cmd.Parameters.AddWithValue("u", username);
cmd.Parameters.AddWithValue("p", hashedPassword);
using (var reader = cmd.ExecuteReader())
{
if (reader.Read())
{
SessionService.UserId = reader.GetInt32(0);
SessionService.Username = reader.GetString(1);
SessionService.IsProfileComplete = reader.GetBoolean(2);
SessionService.Role = reader.GetString(3);
SessionService.CarClass = reader.IsDBNull(4) ? "" : reader.GetString(4);
ShowResult("✅ Успішний вхід!", isError: false);
await Task.Delay(800);
if (SessionService.Role == "Driver")
{
Application.Current!.MainPage = new MenuDrawerPage();
}
else if (SessionService.Role == "Client")
{
Application.Current!.MainPage =
new NavigationPage(new ClientMenuDrawerPage());
}
else
{
ShowResult("❌ Невідома роль користувача", isError: true);
```

```
}
}
}
}
}
}
catch (Exception ex)
{
ShowResult("⚠ Помилка: " + ex.Message, isError: true);
}
}
```

Код реалізація процесу створення замовлення за типом сплати карти:

```
if (_paymentType == "card")
{
paymentDone = false;
int tempOrderId = 0;
try
{
using (var conn = new NpgsqlConnection(connString))
{
conn.Open();
int clientId;
using (var cmd = new NpgsqlCommand(
"SELECT id FROM clients WHERE user_id=@uid", conn))
{
cmd.Parameters.AddWithValue("uid", SessionService.UserId);
clientId = Convert.ToInt32(cmd.ExecuteScalar());
}

string sql = @"
INSERT INTO orders
(client_id, pickup_address, destination_address,
status, car_class, distance_km, price,
payment_type, payment_status)
VALUES (@c, @p, @d, 'Pending', @cl, @dist, @price, 'card', 'pending')
RETURNING id";

using (var cmd = new NpgsqlCommand(sql, conn))
{
cmd.Parameters.AddWithValue("c", clientId);
cmd.Parameters.AddWithValue("p", pickup);
cmd.Parameters.AddWithValue("d", dest);
cmd.Parameters.AddWithValue("cl", _selectedClass);
cmd.Parameters.AddWithValue("dist", dist);
cmd.Parameters.AddWithValue("price", price);
tempOrderId = Convert.ToInt32(cmd.ExecuteScalar());
}
}
}
catch (Exception ex)
{
```

```
await DisplayAlert("Помилка", ex.Message, "OK");
return;
}
await Navigation.PushAsync(new LiqPayPage(price, tempOrderId, async (success) =>
{
if (success)
{
try
{
using (var conn = new NpgsqlConnection(connString))
{
conn.Open();
string sql = @"
UPDATE orders
SET status = 'Active',
payment_status = 'paid'
WHERE id = @oid";
using (var cmd = new NpgsqlCommand(sql, conn))
{
cmd.Parameters.AddWithValue("oid", tempOrderId);
cmd.ExecuteNonQuery();
}
}
_activeOrderId = tempOrderId;
SwitchToTrackingMode(pickup, dest);
}
catch { }
}
else
{           try
{
using (var conn = new NpgsqlConnection(connString))
{
conn.Open();
string sql = "DELETE FROM orders WHERE id = @oid";
using (var cmd = new NpgsqlCommand(sql, conn))
{
cmd.Parameters.AddWithValue("oid", tempOrderId);
cmd.ExecuteNonQuery();
}
}
}
catch { }
}
}));
return;
}
```

Код відображення головної сторінки диспетчер:

```
private async void BuildUI()
{
this.Text = "TaxiSystem — Головна панель";
```

```
this.Size = new Size(1280, 780);
this.StartPosition = FormStartPosition.CenterScreen;
this.BackColor = Color.FromArgb(15, 23, 36);
this.Font = new Font("Segoe UI", 9.5f);
this.MinimumSize = new Size(1280, 780);
var pnlTop = new Panel
{
Dock = DockStyle.Top,
Height = 60,
BackColor = Color.FromArgb(20, 30, 48)
};

var lblTitle = new Label
{
Text = "🚖 TaxiSystem",
Font = new Font("Segoe UI", 18f, FontStyle.Bold),
ForeColor = Color.White,
AutoSize = true,
Location = new Point(20, 14)
};

var lblDate = new Label
{
Text = DateTime.Now.ToString("dd.MM.yyyy  HH:mm"),
Font = new Font("Segoe UI", 9f),
ForeColor = Color.FromArgb(100, 116, 139),
AutoSize = true,
Location = new Point(1050, 22)
};

pnlTop.Controls.AddRange(new Control[] { lblTitle, lblDate });

var pnlNav = new Panel
{
Width = 200,
Dock = DockStyle.Left,
BackColor = Color.FromArgb(20, 30, 48)
};

var lblNavTitle = new Label
{
Text = "НАВІГАЦІЯ",
Font = new Font("Segoe UI", 8f, FontStyle.Bold),
ForeColor = Color.FromArgb(245, 158, 11),
AutoSize = true,
Location = new Point(16, 16)
};
pnlNav.Controls.Add(lblNavTitle);

var navItems = new (string text, string icon, Action action)[]
{
```

```
("Користувачі",        "👥", () => new UsersForm().Show()),
("Замовлення",         "📋", () => new OrdersForm().Show()),
("Створити замовлення", "➕", () => new CreateOrderForm().Show()),
("Активні замовлення",  "🟢", () => new ActiveOrdersForm().Show()),
("Тарифи",             "💰", () => new TariffsForm().Show()),
("Звіт поїздок",       "📊", () => new zvit1().Show()),
("Новий менеджер",     "👤", () => new NewUsersCreated().Show()),
};

int btnY = 50;
foreach (var item in navItems)
{
var btn = new Button
{
Text = $"{item.icon}  {item.text}",
Size = new Size(176, 44),
Location = new Point(12, btnY),
FlatStyle = FlatStyle.Flat,
BackColor = Color.Transparent,
ForeColor = Color.FromArgb(148, 163, 184),
Font = new Font("Segoe UI", 10f),
TextAlign = ContentAlignment.MiddleLeft,
Cursor = Cursors.Hand,
Padding = new Padding(8, 0, 0, 0)
};
btn.FlatAppearance.BorderSize = 0;
var action = item.action;
btn.Click += (s, e) => action();
btn.MouseEnter += (s, e) =>
{ btn.BackColor = Color.FromArgb(30, 41, 59); btn.ForeColor = Color.White; };
btn.MouseLeave += (s, e) =>
{ btn.BackColor = Color.Transparent; btn.ForeColor = Color.FromArgb(148, 163, 184); };
pnlNav.Controls.Add(btn);
btnY += 50;
}

var btnExit = new Button
{
Text = "🚪  Вийти",
Size = new Size(176, 44),
Location = new Point(12, btnY + 10),
FlatStyle = FlatStyle.Flat,
BackColor = Color.Transparent,
ForeColor = Color.FromArgb(239, 68, 68),
Font = new Font("Segoe UI", 10f),
TextAlign = ContentAlignment.MiddleLeft,
Cursor = Cursors.Hand,
Padding = new Padding(8, 0, 0, 0)
};
btnExit.FlatAppearance.BorderSize = 0;
btnExit.Click += (s, e) =>
```

```
{
if (MessageBox.Show("Вийти з системи?", "Вихід",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
Application.Exit();
};
pnlNav.Controls.Add(btnExit);
var pnlContent = new Panel
{
Dock = DockStyle.Fill,
BackColor = Color.FromArgb(15, 23, 36)
};
var pnlStats = new Panel
{
Size = new Size(1050, 110),
Location = new Point(10, 10),
BackColor = Color.Transparent
};

var statsCards = new (string icon, string title, string key, Color accent)[]
{
("🟢", "Активні поїздки",    "active",  Color.FromArgb(34,  197, 94)),
("🚗", "Водіїв онлайн",      "drivers", Color.FromArgb(245, 158, 11)),
("👥", "Клієнтів",           "clients", Color.FromArgb(59,  130, 246)),
("📋", "Замовлень сьогодні", "today",   Color.FromArgb(168, 85,  247)),
};
```

Код реалізації чату спілкування клієнта з водієм або водія з клієнтом:

```
private void LoadMessages()

{

try

{

using (var conn = new NpgsqlConnection(connString))

{

conn.Open();

string sql = @"

SELECT m.id, m.message,

u.username,

TO_CHAR(m.sent_at, 'HH24:MI'),

m.sender_id

FROM messages m

JOIN users u ON m.sender_id = u.id

WHERE m.order_id = @oid

ORDER BY m.sent_at ASC";
```

```
using (var cmd = new NpgsqlCommand(sql, conn))
{
cmd.Parameters.AddWithValue("oid", _orderId);
using (var reader = cmd.ExecuteReader())
{
var items = new List<ChatMessage>();
while (reader.Read())
{
var senderId = reader.GetInt32(4);
var isMine = senderId == SessionService.UserId;
items.Add(new ChatMessage
{
Id = reader.GetInt32(0),
Message = reader.GetString(1),
SenderName = reader.GetString(2),
SentAt = reader.GetString(3),
IsMine = isMine
});
if (reader.GetInt32(0) > _lastMessageId)
_lastMessageId = reader.GetInt32(0);
}
MessagesList.ItemsSource = items;
if (items.Count > 0)
MessagesList.ScrollTo(
items[items.Count - 1],
position: ScrollToPosition.End,
animate: false);
}
}
string markSql = @"
UPDATE messages
SET is_read = true
WHERE order_id = @oid
```

```
AND sender_id != @uid";
using (var cmd = new NpgsqlCommand(markSql, conn))
{
cmd.Parameters.AddWithValue("oid", _orderId);
cmd.Parameters.AddWithValue("uid", SessionService.UserId);
cmd.ExecuteNonQuery();
}
}
}
catch { }
}
```

Додаток В

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Вірянська Дар'я Василівна, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Інформаційна система обліку замовлень послуг таксі» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):

✓ [ ] інструменти генеративного ШІ не використовувалися.

○ [ ] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:

- [ ] перекладу іншомовних джерел;
- [ ] стилістичного редагування та перевірки граматики;
- [ ] допомоги у написанні/відлагодженні програмного коду;
- [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026 р. _____ **ВІРЯНСЬКА ДАР'Я**
(підпис)