ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА
(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем
(повна назва кафедри (предметної, циклової комісії))

Кваліфікаційна робота

на здобуття рівня вищої освіти «бакалавр»
(рівень вищої освіти)

Інформаційна система управління готелем для тварин

(тема кваліфікаційної роботи українською мовою)

Management information system of the animal hotel

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач ВО денної форми навчання
спеціальності 126 – Інформаційні системи та технології .
(шифр і назва напряму підготовки, спеціальності)
Освітня програма «Інформаційні системи та технології»
(назва освітньої програми)
Гавинський Ілля Андрійович
(прізвище, ім'я, по-батькові)

Керівник доктор технічних наук, професор Малахов Є.В.
(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент старший викладач Максимов О.С.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№ 11 від « 11 » червня 2025 р.

Завідувач кафедри

_____ Євгеній МАЛАХОВ
(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № ___
протокол № __ від «___» ____ 2025 р.
Оцінка _____/_____/_____
(за національною шкалою, шкалою ECTS, бали)
Голова ЕК

_____ Олена АРСІРІЙ
(підпис) (ім'я, прізвище)

Одеса - 2025

АНОТАЦІЯ

Мета даного дипломного проекту – проектування і реалізація інформаційної системи готелю для тварин. Користувачами даної системи є асистенти, грумери, завскладом, клієнти, менеджер та ветеринари. Реалізація виконана з використанням мови C# і СУБД PostgreSQL. Архітектура системи відповідає шаблону MVP.

У розробленій інформаційній системі передбачений повний контроль над інформацією щодо сервісів, ресурсів, які використовуються сервісами, резерваціями клієнтів та працівників.

Особливістю системи є можливість ведення повного та детального обліку інформації щодо кожної резервації, такої як обрані сервіси, витрачені на них ресурси, номер зайнятої кімнати та інші. Також для клієнтів та ветеринарів реалізована система відеоспостереження за вихованцями. В інформаційній системі реалізований захист від несанкціонованого доступу та здійснено розмежування повноважень різних категорій користувачів.

Результатом дипломного проектування є інформаційна система управління готелем для тварин зі зручним призначеним для користувача інтерфейсом, що легко сприймається.

ABSTRACT

The purpose of this bachelor's qualification work is to design and implement an information system for a pet hotel. The users of this system are assistants, groomers, warehouse manager, clients, manager and veterinarians. The implementation was done using the C# language and the PostgreSQL database. The system architecture corresponds to the MVP template.

The developed information system provides full control over information about services, resources used by services, customer and employee reservations.

A special feature of the system is the ability to keep a complete and detailed record of information on each reservation, such as selected services, resources spent on them, room number, etc. A video broadcast system for pets is also implemented for clients and veterinarians. The information system provides protection against unauthorized access and differentiates the powers of different categories of users.

The result of this work is an information management system for a pet hotel with a convenient and easy-to-understand user interface.

ЗМІСТ

ВСТУП

Інформаційна система для готелю для тварин є важливим інструментом для ефективного управління та обліку ресурсів цього типу закладу. Управління готелем для тварин включає в себе складні процеси, такі як реєстрація клієнтів та їх тварин, ведення обліку резервацій, контроль за витратами ресурсів та інформацією про наявні сервіси.

Однією з ключових переваг інформаційної системи готелю для тварин є здатність автоматизувати процес реєстрації клієнтів та тварин, включаючи збереження їх особистої інформації, історії резервацій та зареєстрованих тварин [1]. Це допомагає зберігати важливі дані в електронному форматі і спрощує доступ до них у разі необхідності.

Крім того, інформаційна система готелю для тварин має можливість контролю інформації щодо сервісів та доступних у готелі типів тварин, що дозволяє оптимізувати прибуток та витрату ресурсів готелю. Це особливо важливо в умовах підвищеного попиту на послуги готелю під час відпусток та святкових періодів.

Не менш важливою є інтеграція інформаційної системи готелю для тварин з системою контролю інформації про працівників. Це дозволяє забезпечити можливість звільняти та приймати на роботу, назначати надбавки та відрахування та також дозволяє оцінювати та аналізувати ефективність роботи працівників. Інтегрована система відеоспостереження та медичного обслуговування тварин є додатковими, але не менш важливими функціями готелю для тварин.

Узагальнюючи, інформаційна система для готелю для тварин є важливим інструментом для оптимізації управління та надання якісних послуг клієнтам, сприяючи ефективному функціонуванню цього типу закладів.

Мета даного курсового проекту – проектування і реалізація інформаційної системи готелю для тварин. Створювана інформаційна

система покликана допомогти працівникам та клієнтам готелю мати доступ до інформації, не вдаючись до допомоги адміністратора. При цьому:

– працівники отримають універсальний інструмент для виконання своїх робочих обов’язків, таких як складання звітів про витрачені ресурси, підтвердження заявок чи отримання інформації щодо резервацій;

– менеджер отримає можливість контролювати зарплатню та посади працівників, редагувати інформацію щодо ресурсів та сервісів готелю та отримувати більш прозоре уявлення про стан справ завдяки аналітичним функціям;

– клієнти отримають зручний інструмент для реєстрації заявок на бронювання місць для своїх вихованців та також матимуть можливість переглядати інформацію щодо готелю та його сервісів.

Для досягнення зазначеної мети необхідно розв’язати наступні задачі:

1) виконати аналіз предметної області;

2) визначити категорії користувачів і сформулювати їхні вимоги до створюваної інформаційної системи;

3) обрати архітектуру і шаблон проектування створюваної інформаційної системи;

4) спроектувати базу даних;

5) обрати технології та засоби реалізації інформаційної системи;

6) з урахуванням вибраних засобів створити БД, а також клієнтські програми для комп’ютерів та мобільних пристроїв, які забезпечать можливість різним категоріям користувачів ефективно маніпулювати даними предметної області відповідно до їхніх повноважень;

7) забезпечити цілісність і безпеку даних як на рівні БД, так і на рівні застосунку;

8) забезпечити захист системи від несанкціонованого доступу і розмежування повноважень з боку різних категорій користувачів.

1 АНАЛІЗ СИСТЕМ ГОТЕЛІВ ДЛЯ ТВАРИН

1.1 Існуючі ІС управління готелями

У якості аналогів ІС готелю для тварин розглянуто дві системи: Cloudbeds та «Лайка».

Cloudbeds – це хмарна платформа для управління готелями, хостелами та іншими об'єктами розміщення [2]. Хоча ця система не спеціалізована для готелів для тварин, багато її функцій співпадають з функціями спеціалізованих систем, тому її можна розглядати як аналог, аналізуючи її переваги та недоліки. Основними задачами системи Cloudbeds є звіти, аналітика, фінансовий облік та керування бронюваннями та номерами.

Переваги ІС Cloudbeds:

– синхронізація бронювань з Booking.com, Airbnb, Expedia та іншими платформами;

– інтеграція з 300+ каналами продажів, включаючи OTA (Online Travel Agencies), метапошуковики та GDS;

– планувальник прибирання номерів і роботи персоналу.

Недоліки ІС Cloudbeds:

– відсутня вбудована система камер відеоспостереження;

– відсутня гнучка система контролю над ресурсами.

«Лайка» – це спеціалізована інформаційна система для готелів, що приймають тварин [3]. Основними задачами системи «Лайка» є керування бронюваннями, кімнатами для тварин та графіками прогулянок, автоматизація оплати та звітності.

Переваги ІС «Лайка»:

– зовнішнє відеоспостереження (у дворі для вигулу тварин);

– система для реалізації сервісу «Фотосесія»;

– сервіс планування доставки тварини до готелю.

Недоліки ІС «Лайка»:

– відсутність системи медичного догляду за тваринами;

– відсутність розподілу тварин за розмірами;

– відсутність системи відеоспостереження за тваринами для клієнтів.

Узагальнюючи, у ІС Cloudbeds реалізовано багато інтеграцій з іншими популярними сервісами, зв’язаними з фінансовими або планувальними задачами.

Але ця система не спеціалізується на вихованцях та не має глибокої системи управління ресурсами, що зменшує аналітичні можливості системи. ІС «Лайка» реалізує більшість необхідних сервісів, які потрібні готелям для тварин, включаючи такі сервіси як «Фотосесія» та «Доставка тварини до готелю». Але в цій системі ресурси не пов’язані тісно з сервісами, на які вони витрачаються, що знижує кількість корисної інформації, яка використовується для аналізу ефективності роботи готелю. Враховуючи інформацію про розглянуті аналоги, сформульовані переваги та недоліки розробленої ІС.

Переваги:

– відеоспостереження за тваринами у кімнатах для клієнтів;

– підсистема аналізів та догляду за тваринами для ветеринарів;

– підсистеми пов’язані з системою використання ресурсів, що дає можливість розширювати аналітику бізнес процесів.

Недоліки:

– відсутня синхронізація з іншими схожими веб-сервісами;

– відсутня система планування закупівлі ресурсів;

– відсутня кросплатформність на Linux та Mac.

1.2 Постановка задачі на створення інформаційної системи

При аналізі предметної області сформульовані основні завдання, які має вирішувати ІС готелю для тварин:

1) редагування інформації про сервіси, ресурси та допустимі види тварин готелю;

2) управління контрактами, посадами, надбавками та відрахуваннями працівників;

3) управління звітами витрачених та доставлених ресурсів;

4) облік резервацій та інформації про клієнтів, їхніх вихованців, включаючи їхні медичні дані, та відгуків;

5) можливість спостерігати за тваринами у кімнатах готелю через систему відеоспостереження з використанням комп'ютерів або мобільних пристроїв.

Користувачів розробленої ІС розділено на 6 основних груп:

1) менеджер готелю – виконує функції управління персоналом, сервісами, правилами готелю та аналітикою;

2) асистент – працює з клієнтами, надаючи їм інформацію про готель. Займається оформленням та підтвердженням резервацій;

3) грумер – доглядає за тваринами та виконує додаткові функції, відповідно до обраних клієнтом сервісів;

4) завскладом – складає звіти про витраченні ресурси та ресурси, які були доставлені на склад;

5) клієнт – має можливість додавати заявки на резервацію тварин, залишати відгуки та дивитися трансляції своїх вихованців з камер у кімнатах готелю;

6) ветеринар – доглядає за тваринами, які потребують медичного догляду.

Сценарії та задачі всіх користувачів перелічені в додатках А та Б відповідно.

Для наочного представлення бажань користувачів та обмежень у реалізації цих бажань розроблена таблиця Jobs To Be Done (JTBD) (табл. 1.1).

Таблиця 1.1 – JTBD

№	Незадоволені бажання	Обмеження	Каталізатори	Варіанти вибору
1	Бажання мати систему взаємодії з сервісами, ресурсами та допустимими видами тварин	Зв’язати системи ресурсів, сервісів та типів тварин є складною задачею, через те, що зв’язки між цими підсистеми є комплексними, що підвищує ризик колізії	Необхідність мати більше інформації для аналітичних задач та більше контролю над готелем для тварин	CloudBeds: Operations & Finance
2	Бажання мати систему управління контрактами, посадами, надбавками та відрахуваннями працівників	Тривалість трудових договорів та права доступу до даних повинні контролюватися менеджером самостійно, що створює додаткове навантаження на менеджера	Додаткова підсистема управління персоналом, яка пов’язана з основною системою, дає можливість оцінити роботу працівників	CloudBeds: Operations & Finance
3	Бажання мати систему управління звітами витрачених та доставлених ресурсів	Витрачені ресурси повинні бути пов’язаними з витратами на конкретні сервіси, що є складною задачею через те, що кілька типів ресурсів можуть бути витраченими на один сервіс, а в свою чергу кілька сервісів можуть бути одночасно призначеними на одну резервацію	Відслідковування витрати ресурсів є важливою задачею як для аналітики, так і для виявлення випадків крадіжки ресурсів	CloudBeds: Revenue & Analytics

Продовження таблиці 1.1

№	Незадоволені бажання	Обмеження	Каталізатори	Варіанти вибору
4	Бажання мати систему обліку резервацій та інформації про клієнтів, їхніх вихованців та відгуків	Резервації – є основним інформаційним об’єктом для реєстрації заявок клієнтів та підключених сервісів, тому вкрай важливо мати надійні системи збереження інформації щодо клієнтів та їхніх вихованців	Необхідність мати фінансову та аналітичну інформацію про резервації клієнтів, що є одним з найголовніших об’єктів для інформаційної системи готелю для тварин	«Лайка» CloudBeds: Guest Experience
5	Бажання мати систему медичного догляду за тваринами та систему відеоспостереження	Відслідковування історії аналізів та прийому препаратів є важливою та складною задачею, через велику кількість самих аналізів, препаратів та вихованців	Необхідність мати наочне відображення історії хвороби тварини та мати можливість дивитися трансляції з кімнати тварини, аби швидко реагувати на зміни у стані здоров’я вихованця	«Лайка»
6	Бажання клієнтів дивитись на своїх вихованців в онлайн режимі	Система відеоспостереження потребує додаткового налаштування відповідних компонентів у застосунку та додаткових заходів для збереження конфіденційності	Клієнти, для свого спокою, хочуть наочно впевнитися в тому, що їхній вихованець здоровий і щасливий	«Лайка»

2 ПРОЕКТУВАННЯ ІС ГОТЕЛЮ ДЛЯ ТВАРИН

2.1 Архітектура та шаблон проектування

При аналізі структури необхідних додатків для комп'ютерів та мобільних пристроїв обрано дворівневу архітектуру клієнт-сервер.

Дворівнева архітектура – це архітектурна модель, що передбачає в її складі два взаємопов'язані компоненти: клієнт та сервер баз даних (рис. 2.1).

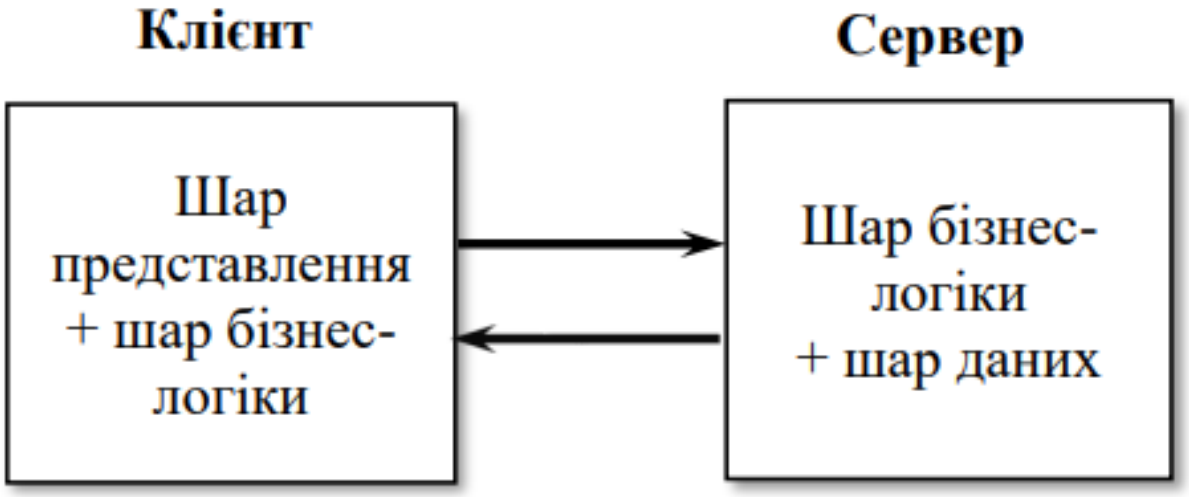


Рисунок 2.1 – Дворівнева архітектура

У цій моделі клієнт відповідає за інтерфейс користувача та обробку локальних даних, тоді як сервер баз даних зберігає та оброблює великі обсяги даних. Ця архітектура дозволяє розділити функціональність між клієнтом і сервером, що спрощує розробку, підтримку та масштабування програмного продукту. Основні характеристики дворівневої архітектури наведені нижче.

Клієнтська частина: клієнтська програма забезпечує зручний інтерфейс для взаємодії з користувачем.

Клієнт може проводити обробку даних на своєму пристрої перед відправкою запитів на сервер. Клієнт взаємодіє з сервером, надсилаючи запити на отримання, змінення або збереження даних.

Серверна частина: сервер баз даних забезпечує централізоване зберігання великих обсягів даних.

Сервер оброблює запити від клієнтів, виконуючи операції з даними, такі як пошук, оновлення, видалення тощо. Також сервер відповідає за керування доступом до даних, забезпечуючи безпеку та конфіденційність інформації.

Дворівнева архітектура обрана для ІС «Готель для тварин» через простоту розробки та легкість масштабування системи. Однак ця архітектура може мати обмеження в масштабуванні та розділенні функціональності між клієнтом і сервером, але цей недолік є незначним для невеликих систем, до яких відноситься ІС «Готель для тварин». У інших аспектах, таких як безпека та швидкість роботи з даними, дворівнева архітектура повністю задовольняє потреби розробленої ІС.

У якості шаблону проектування додатку обрано шаблон MVP, який детально розібраний у роботі [4].

При використанні шаблону MVP, розробленого для поліпшення відділення логіки від відображення, ІС також ділиться на три окремі блоки (рис. 2.2):

1) модель, яка здійснює маніпулювання даними застосунку, надання даних Представнику і реагування на команди Представника шляхом зміни свого стану;

2) представлення, яке відповідає за відображення даних користувачеві (звертаючись при цьому за цими даними до Представника), а також за отримання від користувача команд для роботи з даними і відправку цих команд Представнику;

3) представник, який відповідає за перетворення команд користувача, одержуваних від Представлення, в набір дій над Моделлю з метою її зміни, а також виконує функції зміни Представлення при зміні стану Моделі, будучи своєрідним «посередником» між Моделлю і Представленням.

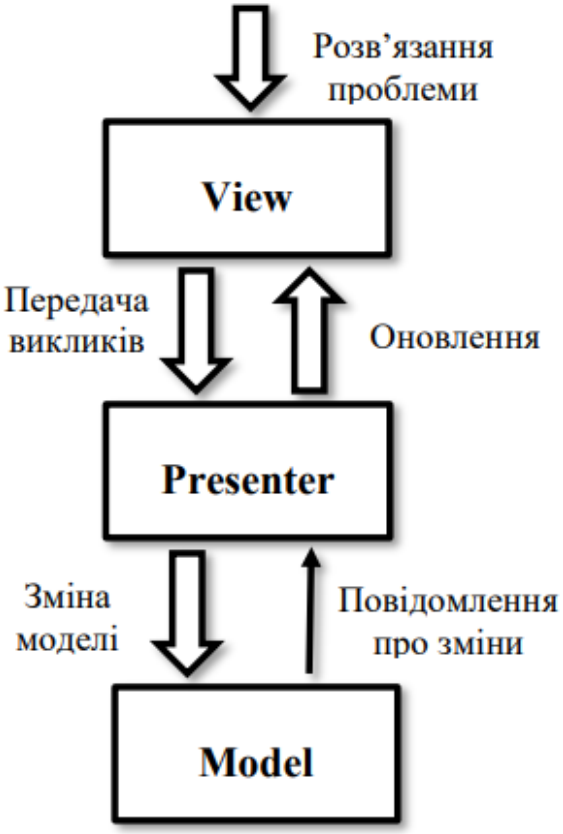


Рисунок 2.2 – Шаблон MVP

Шаблон проектування MVP обраний для ІС «Готель для тварин» через зручне розділення відповідальностей між компонентами програми, що спрощує розробку та тестування, полегшує зміни у логікі програми та підвищує її масштабованість.

Система «Готель для тварин» вимагає як і багатьох незначних за обсягом звертань до БД, так і великих одноразових запитів, які повертають великий обсяг інформації – тому цей шаблон є особливо зручним, порівняно з іншими шаблонами, через те, що він надає легку схему взаємодії компонентів MVP, яка дозволяє швидко розроблювати і виклики подій і їх обробників для незначних запитів до БД, і використання попереднього завантаження даних, які не потребують викликів подій, але все ще підпорядковуються правилам та ідеям шаблону MVP.

На рисунку 2.3 продемонстрована загальна структура взаємодії десктопного та мобільного застосунків, сервера бази даних та системи відеоспостереження [1].

Рисунок 2.3 – Структура ІС

У правій частині рисунка наведена схема взаємодії дестктопного та мобільного застосунків з базою даних.

У кожного з застосунків реалізовані підсистеми взаємодії з сервером бази даних та функцією відеотрансляцій.

У лівій частині рисунку, продемонстрований механізм трансляції відео з камери кімнати тварини у застосунки. Трансляції можуть бути налаштовані у дестктопному застосунку за RTSP протоколом, який є одним з найпопулярніших протоколів для IP-камер, або відео з камер можуть бути попередньо завантажені на YouTube у якості YouTube трансляцій, які потім будуть ретрансльовано до дестктопного та мобільного застосунків.

2.2 Інформаційне моделювання предметної області

Сутності та їх атрибути з описом обмежень, що потрібні для розв’язання поставлених задач, наведено в додатку В.

В базі даних інформаційної системи «Готель для тварин» існують 2 види зв’язку між сутностями, а саме:

1) один-до-багатьох;

2) багато-до-багатьох.

Розглянемо кожний зі зв’язків по черзі.

1) Зв’язок один-до-багатьох формалізується додаванням первинного ключа сутності, що відповідає одиничному зв’язку у якості зовнішнього ключа до сутності зі сторони множинного зв’язку. Такий вид зв’язку наявний між таблицями:

а) Resourse → ProcurementReport;

б) AllowedPetType → PetRoom;

в) PetKind → AllowedPetType;

г) Pet → Reservation;

д) PetRoom → Reservation;

е) Client → Review;

ж) Staff → Reservation;

з) StaffContract → SalaryChange;

и) Pet → Analysis;

к) AnalysisPetType → Analysis.

2) Зв’язок багато-до-багатьох формалізується створенням додаткової сутності та доданням первинних ключів обох сутностей до неї. Подібний зв’язок наявний між наступними сутностями:

а) Resourse ↔ ServicePet, для формалізації яких створена сутність ServicePetResourse, яка виступає окремою логічною сутністю «Використання даного ресурсу даним сервісом для даного типу тварини»;

б) Service ↔ PetKind, для формалізації яких створена сутність ServicePet, яка виступає окремою логічною сутністю «Доступність даного сервісу для даного типу тварини»;

в) ServicePet ↔ Reservation, для формалізації яких створена сутність ReservService, яка виступає окремою логічною сутністю «Реєстрація сервісу до резервації»;

г) Resourse ↔ ReservService, для формалізації яких створена сутність ServiceCostReport, яка виступає окремою логічною сутністю «Фактична витрата ресурсу на сервіс»;

д) AllowedPetType ↔ Client, для формалізації яких створена сутність Pet, яка виступає окремою логічною сутністю «Вихованець клієнта»;

е) Position ↔ Staff, для формалізації яких створена сутність StaffContract, яка виступає окремою логічною сутністю «Прийом працівника на роботу»;

ж) AnalysisType ↔ AllowedPetType, для формалізації яких створена сутність AnalysisPetType, яка виступає окремою логічною сутністю «Нормативне значення даного типу аналізів для даного типу тварин».

Наприклад, розглянемо зв’язок між сутностями Resourse та ServicePet. Кожний ресурс може використовуватися у багатьох сервісів, при цьому кожен сервіс може використовувати більше одного ресурсу, тобто маємо зв’язок багато-до-багатьох.

Наступний приклад, розглянемо зв’язок між сутностями Pet та Reservation. Кожний вихованець може мати безліч резервацій, при цьому кожна резервація буде зареєстрована лише на одного конкретного вихованця. Отже маємо зв’язок один-до-багатьох.

Усі вказані таблиці знаходяться у формі Бойса-Кодда.

ER-діаграма отримана в результаті формалізації зв’язків ІС «Готель для тварин» приведена на рисунку 2.4. Для її побудування використано програмний застосунок diagrams.net з використанням нотації «вороняча лапка».

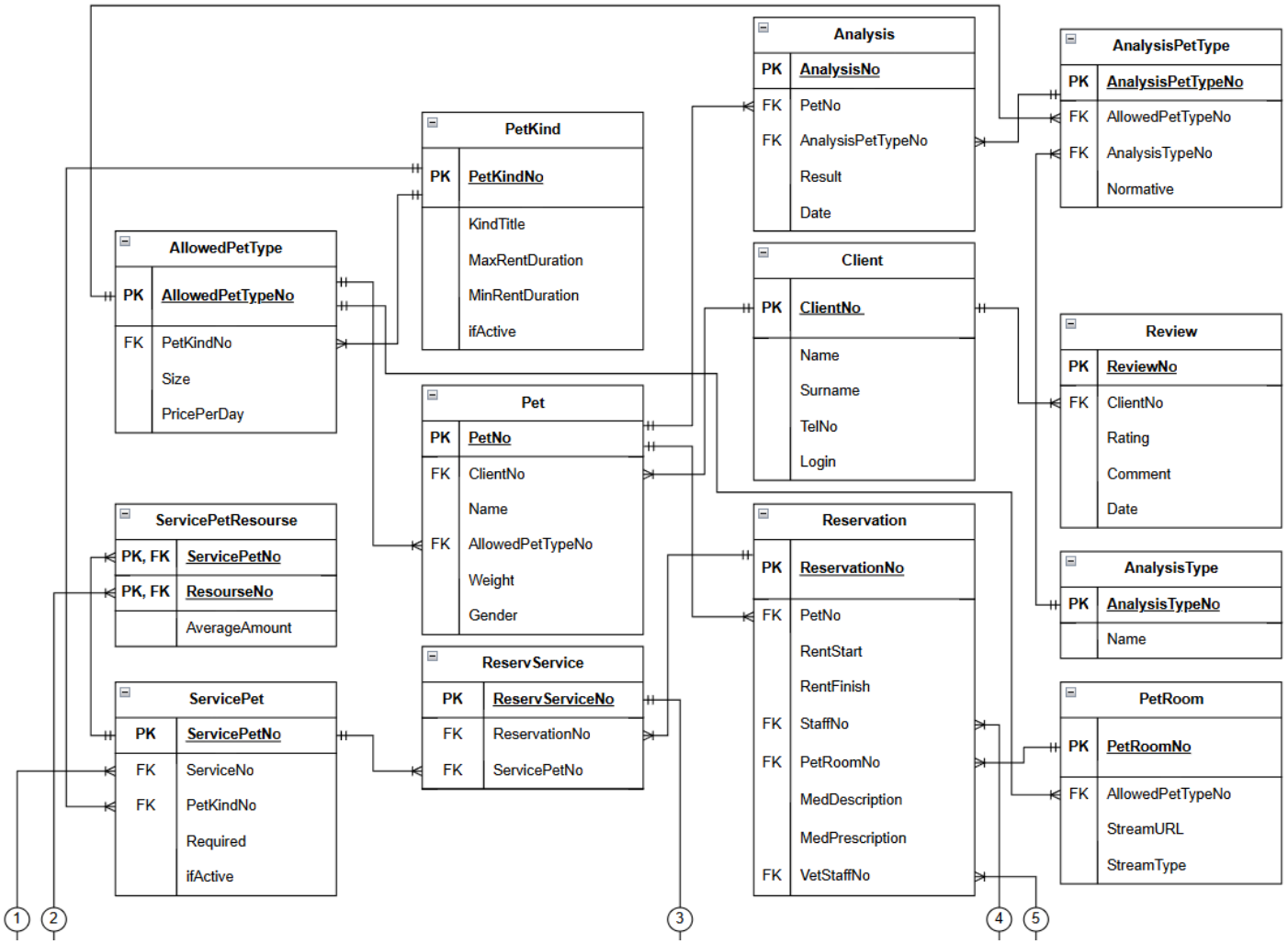
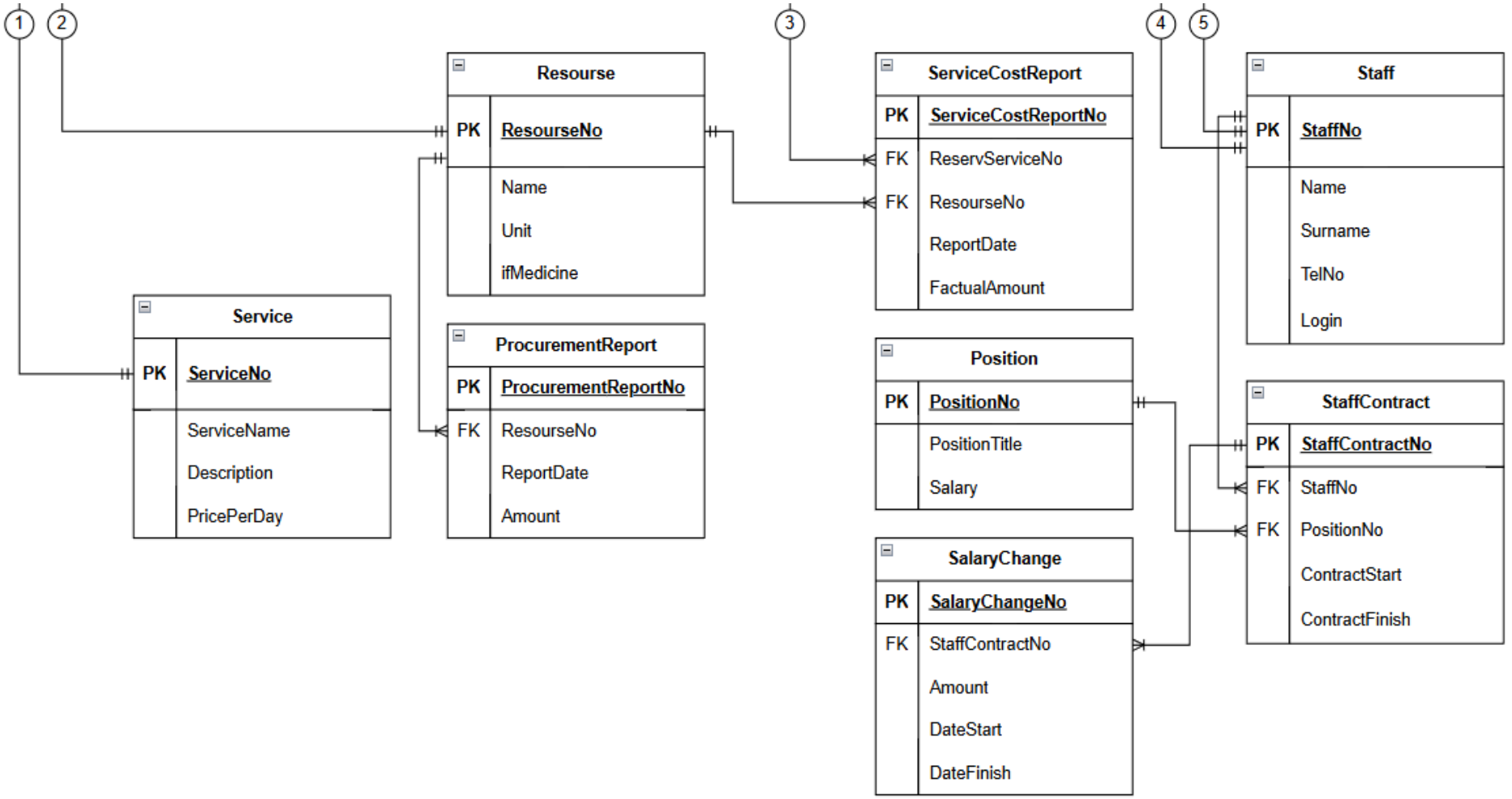


Рисунок 2.4 – ER-діаграма інформаційної системи «Готель для тварин»



Продовження рисунку 2.4

2.3 Програмна модель дестктопного застосунку

Модель взаємодії застосунку з сервером бази даних реалізована використовуючи шаблон MVP. Застосунок має 6 форм, по одної для кожної ролі користувачів та одна для логіну та реєстрації, які являють собою інтерфейси з якими взаємодіє користувач. Тобто, для кожної форми є свій клас інтерфейсу, який вона реалізує та представник, який посилається та взаємодіє з формою через інтерфейс. Представник у свою чергу посилається на єдиний для всіх представників клас моделі, який вже виконує з'єднання та проводить операції з сервером БД.

Перебіг обміну інформації між застосунком та сервером бази даних можна описати таким чином: після взаємодії користувача з компонентом форми, викликається відповідна подія, на яку, вже був підписаний представник, так як він посилається на інтерфейс, який реалізований формою.

Представник передає управління обробнику події, який, за необхідністю, перевіряє вхідні дані, та передає їх у відповідний метод моделі. Модель приймає вхідні дані, якщо такі є, посилає запит до серверу бази даних та повертає представнику результат операції. Тепер представник оброблює результат роботи метода моделі, та виконує відповідні дії, змінюючи та оновлюючи властивості форми або повідомляючи користувача о помилці та причинах її виникнення.

Кожні з 7 пар представник-інтерфейс є повністю аналогічними за своєю структурою, та відрізняються тільки кількістю та типами властивостей, методів та подій. В якості прикладу наведена діаграма зв'язків між формою клієнта, її представником та моделлю (рис. 2.5).

На рисунку інтерфейс `IViewClientForm` відігріє роль Представлення, `ClientForm` реалізує цей інтерфейс, клас `ClientFromPresenter` відіграє роль Представника, а клас `Model` відіграє роль моделі у шаблоні MVP.

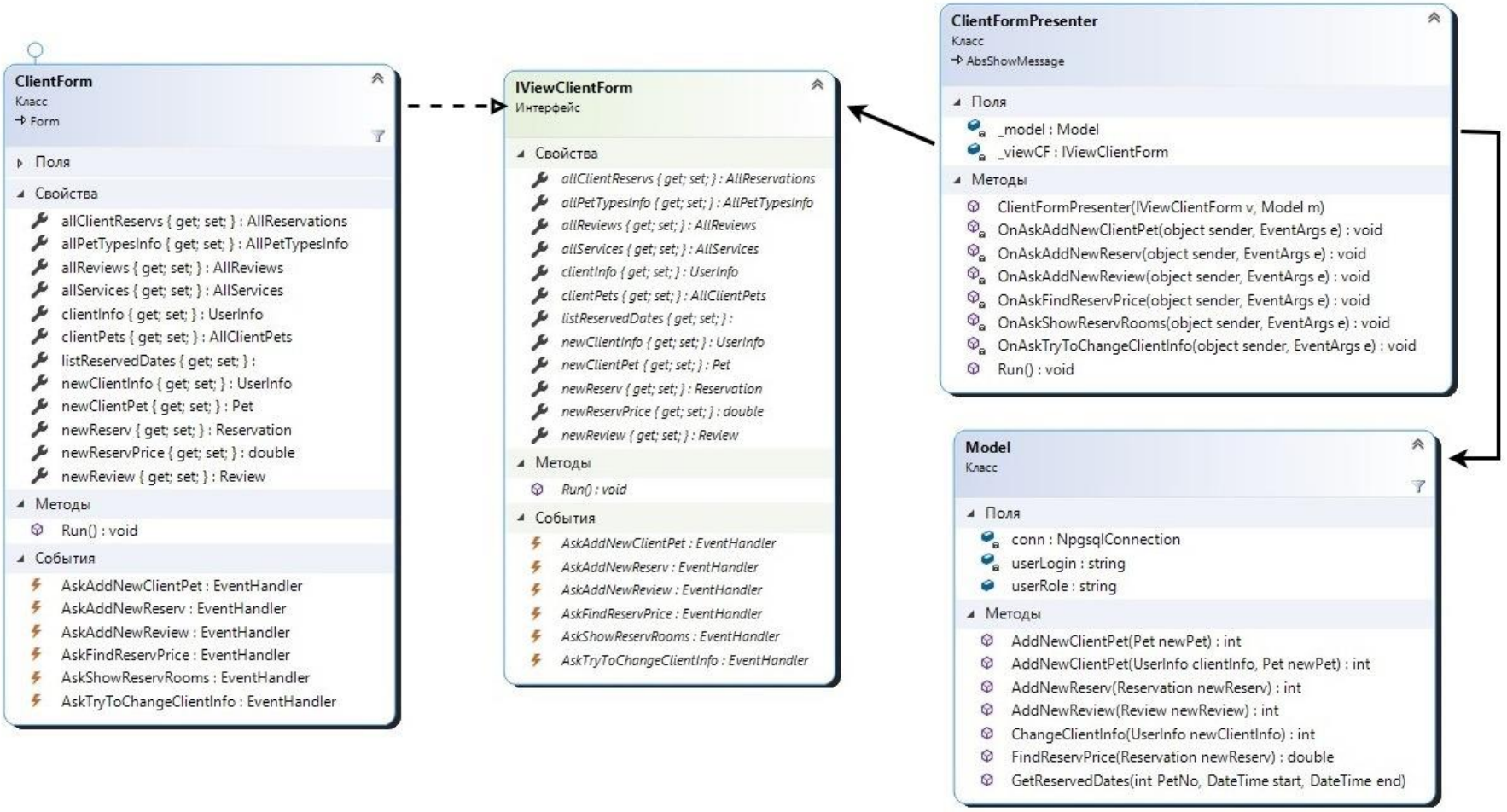


Рисунок 2.5 – Діаграма прикладу зв’язків між класами

Як видно на рисунку 2.5, форма, інтерфейс, представлення та модель часто використовують у якості властивостей чи аргументів методів спеціальні класи, які у той чи іншій степені відповідають таблицям чи комбінаціям таблиць бази даних. Загальний вигляд таких спеціальних класів продемонстровано на прикладі таблиці SalaryChange (рис. 2.6), яка транслюється у C# в однойменний клас.

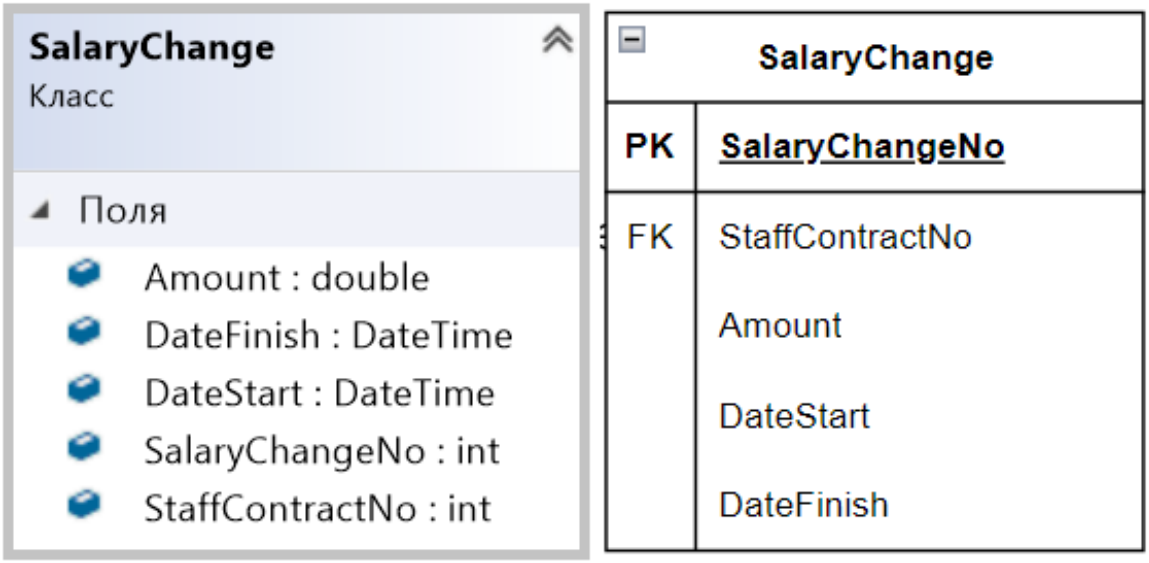


Рисунок 2.6 – Порівняння таблиці БД та відповідного класу застосунку

2.4 Програмна модель мобільного застосунку

Структура мобільного застосунку підпорядковується правилам шаблону MVP та має схожу реалізацію з десктопним застосунком, але має деякі відмінності, через значно менший об’єм функцій, які повинні бути реалізовані та через іншу платформу програмування.

По перше, в мобільному застосунку реалізований такий обмежений набір функцій та можливостей:

– можливість авторизації тільки для клієнтів та ветеринарів;

– клієнти можуть переглядати трансляції своїх вихованців, які знаходяться в готелі на поточний момент;

– ветеринари можуть переглядати трансляції вихованців, які потребують медичного догляду;

– підтримка тільки YouTube трансляцій.

По друге, всього застосунок має 3 сторінки (які в рамках платформи .NET MAUI називаються «page»): сторінки авторизації, клієнта та ветеринара.

Розробка компонентів та зовнішнього вигляду сторінок виконується використовуючи технологію XML.

Через особливості створення та налаштування механізму сторінок на цій платформі, шаблон MVP спрощений до представлення та моделі, тобто, сторінки звертаються безпосередньо до статичного класу моделі. Також відсутні допоміжні класи так як об’єм інформації, який передається до застосунку від бази даних, є дуже малим та не потребує складної структуризації.

Метод взаємодії з сервером бази даних у мобільному застосунку є таким самим як і в дестктопному застосунку – через методи моделі, використовуючи бібліотеку npgSQL.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІС ГОТЕЛЮ ДЛЯ ТВАРИН

3.1 Вибір програмного забезпечення

ІС готелю для тварин розроблена у форматі дестктопного та мобільного застосунку. Система реалізована с залученням наступних технологій:

1) обрана база даних – СУБД PostgreSQL;
2) мова програмування дестктопного та мобільного застосунку – C#;
3) середовище розробки Visual Studio 2019 Professional Edition;
4) модулі інтерфейсу Windows Forms для дестктопного застосунку;
5) фреймворк .NET MAUI для мобільного застосунку;
6) бібліотека класів Npgsql для роботи з СУБД PostgreSQL;
7) бібліотека класів libVLCSharp для роботи з відео трансляціями.

СУБД PostgreSQL [5] обрано через декілька основних причин:

1) дана СУБД є вільно поширюваною та безкоштовною;
2) наявна велика кількість документації у відкритому доступі;
3) об’єктно-реляційна модель представлення даних дозволяє легко уявити у класи мови об’єктного-орієнтованого програмування C#.

Visual Studio обраний у якості середовища розробки через його ефективність та зручність відображення сутностей у вигляді класів, та реалізації необхідних операцій завдяки зручним інструментам для використовування ООП.

Для представлення інформації користувачу у дестктопному застосунку обрані модулі інтерфейсу Windows Forms, які детально розібрані у роботі [6]. Основними причинами вибору цих модулів у якості інтерфейсу є їхня гнучкість у редагуванні та додавання нових компонентів форми, а також швидка та проста взаємодія з іншими класами за допомогою механізму подій.

Для розробки мобільного застосунку обрана платформа .NET MAUI [7]. Основними перевагами цієї платформи є висока інтегрованість з

іншими компонентами .NET, що значно спрощує тестування та розширення системи у рамках середовища розробки Visual Studio. .NET MAUI надає можливість тестування застосунків на різних мобільних пристроїв з можливістю налаштування параметрів у вбудованому емулятору Android пристроїв. Також є можливість розробляти кросплатформні застосунки, які, маючи єдиний код, можуть запускатися на різних платформах, що надає ширші можливості для подальшого розвитку ІС готелю для тварин.

Бібліотека класів Npgsql для роботи з СУБД PostgreSQL обрана через її популярність, а отже, і наявність великої кількості документації по цій бібліотеці. Методи та функції класів бібліотеки Npgsql є достатніми для виконання поставлених задач, пов’язаних з операціями з БД – зайва складність та обсяг функцій сповільнили б взаємодію застосунку да сервера бази даних.

Бібліотека класів libVLCSharp для роботи з відео трансляціями обрана через можливість роботи з більшістю форматів відео та високою кількістю налаштувань відеоплеєру [8]. Для налаштування YouTube трансляції достатньо компонента WebView2 з платформи WinForms, але для RTSP трансляцій потрібна окрема бібліотека, якою і стала бібліотека libVLCSharp. Значною перевагою цієї бібліотеки також є те, що вона надає пакет спеціально розроблений для інтеграції з платформою . NET.

3.2 Створення бази даних

Нижче наведено приклади запитів на створення таблиць у базі даних інформаційної системи «Готель для тварин».

Детально розглянемо запит на створення таблиці PetKind. Для створення таблиці використовується наступний SQL-запит (лістинг 3.1).

```
create table PetKind (
    PetKindNo serial not null
```

Лістинг 3.1 – Запит на створення таблиці `PetKind`

```
    primary key,
KindTitle varchar not null
    unique,
MaxRentDuration int not null
    check (MaxRentDuration > 0
          and MaxRentDuration >= MinRentDuration),
MinRentDuration int not null
    check (MinRentDuration > 0
                and MinRentDuration <= MaxRentDuration),
ifActive bool not null
    default TRUE
);
```

Лістинг 3.1, аркуш 2

Створення таблиці відбувається за допомогою команди `CREATE TABLE`, далі вказується ім'я таблиці. Кожному полю таблиці зіставляються тип даних обраної СУБД (PostgreSQL), яка детально розібрана у роботі [9], і обмеження цілісності.

Поле `PetKindNo` є первинним ключем (`PRIMARY KEY`) і має тип `serial` (цей тип не є справжнім типом даних, а являє собою зручний спосіб створення унікального ідентифікатора шляхом збільшення попереднього значення в даному стовпці на вказане значення, за замовчуванням на 1). Поле `matrix` має тип `varchar`, який позначає текстовий формат даних. `NOT NULL` при визначенні полів таблиці означає, що вони не можуть містити порожні значення. Запис `unique` при визначенні поля `KindTitle` означає, що значення у цьому полі має бути унікальним серед всіх значень аналогічних полів інших записів цієї таблиці. `check(MaxRentDuration > 0 and MaxRentDuration >= MinRentDuration)`, при визначенні поля `MaxRentDuration` вказує, що значення у цьому полі має бути більше нуля та більше або дорівнювати значенню поля `MinRentDuration`. Запис `DEFAULT true` при визначенні поля `ifActive` типу `boolean` означає, що дані у полі `ifActive` за замовчуванням приймають значення `true`, тим самим позначаючи стан цього типу тварини як доступний для резервації.

Розглянемо ще два запити на створення таблиць (лістинг 3.2).

```
create table Client (
    ClientNo serial not null
        primary key,
    Name varchar not null,
    Surname varchar not null,
    TelNo varchar not null
        check
  (TelNo similar to '0[0-9]{2}-[0-9]{3}-[0-9]{2}-[0-9]{2}')
        unique,
    Login varchar not nul
        unique
    );
create table Review (
    ReviewNo serial not null
        primary key,
    ClientNo int not null
        references Client on update cascade,
    Rating smallint not null
        check (Rating between 1 and 5),
    Comment varchar,
    Date date not null
        check(Date <= current_date)
        default current_date
    );
```

Лістинг 3.2 – Запити на створення таблиць `Client` та `Review`

Перший з них демонструє створення батьківської таблиці Client, а другий – створення дочірньої таблиці Review, в якій поле ClientNo визначене як зовнішній ключ або посилання (REFERENCES) з метою забезпечення зв'язку з полем ClientNo таблиці Client.

Тексти програм SQL-запитів, які використовуються для створення всіх таблиць БД, наведені в додатку Г.

SQL-запити представлень та користувацьких функцій наведені у додатках Д і Е відповідно.

3.3 Запити до бази даних для розв’язання задач користувачів ІС

Для вирішення задач інформаційної системи «Готель для тварин» необхідні різноманітні SQL-запити, функції та представлення. Нижче наведений номер кожної задачі та метод її вирішення.

1) Для вирішення задач М1, М19, М24, М26, М30, М32, А1, А2, А3, Г1, Г2, Г3, Зв1, Зв2, Зв3, А5, К5, А6, А10, Г5, Зв5, К4, К6, К7, В1, В2, В3, В9, В10, В11 виконується запит типу `SELECT список_стовпців FROM таблиця/представлення [WHERE умова]` з підстановкою відповідних імен таблиць/представлень, списку стовпців та умов, коли вони необхідні:

– для задачі М1 – ім'я представлення `AllPetTypesInfo_Manag` (див. додаток Д) список стовпців – всі;

– для задачі М1 – ім'я таблиці `Service` список стовпців – всі;

– для задачі М1 – ім'я представлення `AllServicePets_Manag` (див. додаток Д) список стовпців – всі;

– для задачі М19 – ім'я представлення `AllStaffInfo_Manag` (див. додаток Д) список стовпців – всі;

– для задачі М19 – ім'я представлення `AllStaffContracts_Manag` (див. додаток Д) список стовпців – всі, за умови збігу `Login`;

– для задачі М19 – ім'я представлення `StaffSalaryChanges_Manag` (див. додаток Д) список стовпців – всі, за умови збігу `StaffContractNo`;

– для задачі М24 – ім'я таблиці `Position` список стовпців – `PositionTitle, Salary`, за умови `PositionTitle <> 'Manager'`;

– для задачі М26 – ім'я представлення `Analitic_ClientRating` (див. додаток Д) список стовпців – всі;

– для задачі М30 – ім'я представлення `Analitic_ProcReports` (див. додаток Д) список стовпців – всі, за умови `ReportDate` знаходиться у обраному користувачем діапазоні дат;

– для задачі М32 – ім'я представлення `Analitic_ClientComments` (див. додаток Д) список стовпців – всі, за умови `Date` знаходиться у обраному користувачем діапазоні дат;

– для задач А1, Г1, Зв1, В1 – ім'я представлення `StaffInfo` (див. додаток Д) список стовпців – всі;

– для задач А2, Г2, Зв2, В2 – ім’я представлення `StaffContracts` (див. додаток Д) список стовпців – всі;

– для задач А3, Г3, Зв3, В3 – ім’я представлення `StaffSalaryChanges` (див. додаток Д) список стовпців – всі;

– для задач А5, К5 – ім’я представлення `AllServicePets` (див. додаток Д) список стовпців – всі;

– для задач А5, К5 – ім’я представлення `AllPetTypesInfo` (див. додаток Д) список стовпців – всі;

– для задачі А6 – ім’я представлення `AllClientReservs` (див. додаток Д) список стовпців – всі;

– для задачі А10 – ім’я таблиці `Client` список стовпців – `Name, Surname, TelNo,` за умови `Login` дорівнює обраному асистентом логіну;

– для задачі Г5 – ім’я представлення `Pets_Groomer` (див. додаток Д) список стовпців – всі;

– для задачі Зв5 – ім’я представлення `ResStorage_WareHM` (див. додаток Д) список стовпців – `Name, Amount, Unit`;

– для задачі К4 – ім’я представлення `ClientPets` (див. додаток Д) список стовпців – всі;

– для задачі К6 – ім’я представлення `AllReviews_Client` (див. додаток Д) список стовпців – всі;

– для задачі К7 – ім’я представлення `ClientReservs` (див. додаток Д) список стовпців – всі;

– для задачі В9 – ім’я представлення `AllReservs_Vet` (див. додаток Д) список стовпців – всі;

– для задачі В10 – ім’я представлення `AllMedResourses_Vet` (див. додаток Д) список стовпців – всі, за умови `PetNo` дорівнює обраному ветеринаром номеру вихованця;

– для задачі В11 – ім’я таблиці `Analysis`, список стовпців – всі, за умови `PetNo` дорівнює обраному ветеринаром номеру вихованця.

2) Для вирішення задач К2, А4, Г4, Зв4, М3, М4, М11, М23, М25, В4, В6, В8 виконується запит типу `UPDATE таблиця/представлення SET стовпець_1 = значення_1 […, стовпець_n = значення_n] [WHERE умова]` з підстановкою відповідних імен таблиць та списку необхідних стовпців:

– для вирішення задачі К2 – ім’я представлення `clientinfo`, список стовпців – `name`, `surname`, `telno`, умова – співпадіння відсутня (див. додаток Д);

– для вирішення задач А4, Г4, Зв4, В4 – ім’я представлення `staffinfo`, список стовпців – `name`, `surname`, `telno`, умова – співпадіння відсутня (див. додаток Д);

– для вирішення задачі М3 – ім’я таблиці `ServicePet`, список стовпців – `ifActive`, умова – співпадіння `ServiceNo`;

– для вирішення задачі М4 – ім’я таблиці `Service`, список стовпців – `ServiceName`, `Description`, `PricePerDay`, умова – співпадіння `ServiceNo`;

– для вирішення задачі М11 – ім’я таблиці `Resourse`, список стовпців – `Name`, `Unit`, умова – співпадіння `Name`;

– для вирішення задачі М23 – ім’я представлення `StaffContract`, список стовпців – `ContractFinish`, умова – співпадіння `StaffContractNo` (див. додаток Д);

– для вирішення задачі М25 – ім’я таблиці `Position`, список стовпців – `Salary`, умова – співпадіння `PositionTitle`;

– для вирішення задачі В6 – ім’я таблиці `AnalysisType`, список стовпців – `Name`, умова – співпадіння `AnalysisTypeNo`;

– для вирішення задачі В8 – ім’я таблиці `AnalysisPetType`, список стовпців – `Normative`, умова – співпадіння `AnalysisPetTypeNo`.

3) Для вирішення задач М12, М17, В7 виконується запит типу `DELETE FROM` таблиця `WHERE id_таблиці = значення [додаткова умова]` з підстановкою відповідних імен таблиць та стовпців:

– для вирішення задачі М12 – ім’я таблиці `Resourse`, умова – співпадіння `Name`;

– для вирішення задачі М17 – ім’я таблиці `PetRoom`, умова – співпадіння `PetRoomNo`;

– для вирішення задачі В7 – ім’я таблиці `AnalysisType`, умова співпадіння `AnalysisTypeNo`.

4) Для вирішення задач ХХ створені користувацькі функції, звернення до яких реалізується через запит типу `select * from ім’я_функції([параметр 1], [параметр 2], [параметр 3], […])`, з підстановкою відповідної назви функції та значень параметрів, якщо такі є. Функції можуть повертати або одиничне значення, або кортеж чи кілька кортежів таблиці, або нульове значення. У тіло функції ми можемо додати будь які перевірки та операції обробки інформації, якщо це необхідно. Саме з цієї причини, функції є достатньо універсальним та зручним засобом для вирішення більшості задач користувачів:

– для вирішення задачі М2 – ім’я функції `AddNewService_Manag` (див. додаток Е.15), параметри функції – назва сервісу, опис сервісу, ціна;

– для вирішення задач М5, М6 – ім’я функції `AddEditTypeServ_Manag` (див. додаток Е.14), параметри функції – назва сервісу, тип тварини;

– для вирішення задач М7, М8 – ім’я функції `AddEditServPetRes_Manag` (див. додаток Е.17), параметри функції – назва сервісу, ресурсу та тварини, кількість ресурсу;

– для вирішення задачі М9 – ім’я функції `DeleteServPetRes_Manag` (див. додаток Е.18), параметри функції – назва сервісу, ресурсу та тварини;

– для вирішення задачі М13 – ім’я функції `AddNewPetType` (див. додаток Е.21), параметри функції – назва, мінімальна та максимальна тривалість резервації, ціна за день для трьох розмірів тварини;

– для вирішення задач М14, 15 – ім’я функції `EditPetType` (див. додаток Е.20), параметри функції – назва, мінімальна та максимальна тривалість резервації, ціна за день для трьох розмірів тварини;

– для вирішення задачі М16 – ім’я функції `AddNewRoom` (див. додаток Е.22), параметри функції – тип кімнати, розмір тварин, які можуть перебувати у кімнаті;

– для вирішення задачі М18 – ім’я функції `StaffReg_Manag` (див. додаток Е.27), параметри функції – ім’я, прізвище, номер телефону, логін, пароль;

– для вирішення задачі М20 – ім’я функції `AddStaffContract_Manag` (див. додаток Е.24), параметри функції – табельний номер працівника, посада, початок та кінець дії договору;

– для вирішення задачі М22 – ім’я функції `AddSalChange_Manag` (див. додаток Е.23), параметри функції – табельний номер працівника, кількість надбавки чи вирахування, початок та кінець дії надбавки чи вирахування;

– для вирішення задачі М27 – ім’я функції `Analitic_MonthProfit` (див. додаток Е.28), параметри функції – Рік;

– для вирішення задачі М28 – ім’я функції `Analitic_PopularServices` (див. додаток Е.29), параметри функції – Діапазон дат для перегляду;

– для вирішення задачі М29 – ім’я функції `Analitic_ServCostReport` (див. додаток Е.30), параметри функції – Діапазон дат для перегляду;

– для вирішення задачі М31 – ім’я функції `Analitic_StaffReserv` (див. додаток Е.31), параметри функції – Діапазон дат для перегляду;

– для вирішення задачі А7 – ім’я функції `ReservClientInfo_Assist` (див. додаток Е.6), параметри функції – Номер резервації;

– для вирішення задачі А8 – ім’я функції `ConfirmReserv_Assist` (див. додаток Е.7), параметри функції – Номер заявки;

– для вирішення задачі А9 – ім’я функції `DelReserv_Assist` (див. додаток Е.8), параметри функції – Номер заявки;

– для вирішення задачі А11 – ім’я функції `ClientReg` (див. додаток Е.1), параметри функції – Ім’я, прізвище, номер телефону, логін, пароль;

– для вирішення задачі А12 – ім’я функції `PetReg_Assist` (див. додаток Е.9), параметри функції – Ім’я, тип, розмір, вага тварини та логін клієнта;

– для вирішення задачі А13 – ім’я функції `ReservReg` (див. додаток Е.2), параметри функції – Номер вихованця, початок та кінець резервації, номери обраних додаткових сервісів;

– для вирішення задачі Г6 – ім’я функції `ServiceResourses_Groomer` (див. додаток Е.10), параметри функції – Номер резервації;

– для вирішення задачі Зв6 – ім’я функції `CostReport_WareHM` (див. додаток Е.12), параметри функції – Номер резервації, назва сервісу, назва ресурсу, кількість ресурсу;

– для вирішення задачі Зв7 – ім’я функції `CostReport_WareHM` (див. додаток Е.13), параметри функції – Назва ресурсу, кількість ресурсу;

– для вирішення задачі К3 – ім’я функції `PetReg` (див. додаток Е.9), параметри функції – Ім’я, тип, розмір, вага тварини;

– для вирішення задачі К8 – ім’я функції `ReservReg` (див. додаток Е.2), параметри функції – Номер вихованця, початок та кінець резервації, номери обраних додаткових сервісів;

– для вирішення задачі К9 – ім’я функції `AddReview` (див. додаток Е.4), параметри функції – Оцінка, коментар, дата відгуку;

– для вирішення задачі В5 – ім’я функції `AddAnalysisType_Vet` (див. додаток Е.34), параметри функції – Назва типу аналізу;

– для вирішення задачі В8 – ім’я функції `AddAnalysisPetType_Vet` (див. додаток Е.35), параметри функції – Назва типу аналізу, тип та розмір тварини, нормативне значення;

– для вирішення задачі В12 – ім’я функції `AssignRreserv_Vet` (див. додаток Е.32), параметри функції – Номер резервації;

– для вирішення задачі В13 – ім’я функції `EditMedPrescription_Vet` (див. додаток Е.33), параметри функції – Номер резервації, призначене лікування;

– для вирішення задачі В14 – ім’я функції `AddAnalysis_Vet` (див. додаток Е.36), параметри функції – Номер вихованця, номер відношення типу аналізу і типу тварини, результат аналізу.

3.4 Десктопний застосунок

Застосунок написаний у шаблоні MVP, тому всі основні операції будуть виконуватися між або всередині представлень, представників та моделі.

Усього є шість пар представлень та представників, які виконують функції відповідно шести груп користувачів: менеджер, асистент, грумер, завскладом, клієнт та ветеринар. Ще одна пара відноситься до форми логіну та реєстрації користувачів.

У ролі представлень виступають форми Windows Forms, які і є прикладним інтерфейсом, з котрим взаємодіє користувач. Вони реалізують абстрактні інтерфейси, на які вже посилаються представники, таким чином встановлюючи зв'язок з формою. Приклад такого інтерфейсу наведено у лістингу 3.3 – це є інтерфейс `IViewClientForm`, який реалізується формою клієнта.

```
public interface IViewClientForm
    {
        UserInfo clientInfo { get; set; }
        UserInfo newClientInfo { get; set; }
        AllPetTypesInfo allPetTypesInfo { get; set; }
        AllServices allServices { get; set; }
        AllClientPets clientPets { get; set; }
        AllReservations allClientReservs { get; set; }
        AllReviews allReviews { get; set; }
        Pet newClientPet { get; set; }
        List<(int, List<(DateTime, DateTime)>)>
listReservedDates { get; set; }
        Reservation newReserv { get; set; }
        double newReservPrice { get; set; }
        Review newReview { get; set; }
        void Run();
        event EventHandler AskTryToChangeClientInfo;
        event EventHandler AskAddNewClientPet;
        event EventHandler AskAddNewReserv;
        event EventHandler AskFindReservPrice;
        event EventHandler AskAddNewReview;
        event EventHandler AskShowReservRooms;
    }
```

Лістинг 3.3 – Інтерфейс `IViewClientForm` форми клієнта

Властивості інтерфейсу слугують зручним способом збереження даних, завдяки допоміжним класам, які в той чи іншій степені повторюють схему відношень відповідних таблиць у БД.

Приклад такого допоміжного класу наведено у лістингу 3.4 – це є інтерпретований на мову C# клас `Review`, який відображає однойменну таблицю у базі даних.

```
public class Review
    {
        public int ReviewNo;
        public int ClientNo;
        public int Rating;
        public string Comment;
        public DateTime Date;
        public Review(string Rating, string Comment, string Date)
        {
            this.ReviewNo = -1;
            this.ClientNo = -1;
            this.Rating = Convert.ToInt32(Rating);
            this.Comment = Comment;
            this.Date = DateTime.Parse(Date);
        }
        public Review(int Rating, string Comment, DateTime Date)
        {
            this.ReviewNo = -1;
            this.ClientNo = -1;
            this.Rating = Rating;
            this.Comment = Comment;
            this.Date = Date;
        }
        public bool CheckComment()
        {
            bool ifPass = Comment.Contains('(') ||
Comment.Contains(')') || Comment.Contains('\'') ||
Comment.Contains('{') || Comment.Contains('}');
            return !ifPass;
        }
    }
```

Лістинг 3.4 – Допоміжний клас Review

Різні конструктори використовуються для різних цілей та можуть бути як методом автоматичного заповнення невикористаних полів значенням за замовченням, так і просто зручним методом автоматичної конвертації вхідних параметрів конструкторів у потрібний для поля тип значення. Також, у конкретно даному прикладі класу `Review`, є додаткова допоміжна функція `CheckComment()`, яка використовується для перевірки строкових властивостей класу на наявність заборонених символів.

Також властивості класу форми необхідні для передачі даних до свого представника. На лістингу 3.5 наведена реалізація інтерфейсу `IViewClientForm` формою `ClientForm`.

```
public partial class ClientForm : Form, IViewClientForm
    {
        #region Реалізація інтерфейсу
        public UserInfo clientInfo { get; set; }
        public UserInfo newClientInfo { get; set; }
        public AllPetTypesInfo allPetTypesInfo { get; set; }
        public AllServices allServices { get; set; }
        public AllClientPets clientPets { get; set; }
        public AllReservations allClientReservs { get; set; }
        public AllReviews allReviews { get; set; }
        public Pet newClientPet { get; set; }
        public List<(int, List<(DateTime, DateTime)>)>
listReservedDates { get; set; }
        public Reservation newReserv { get; set; }
        public double newReservPrice { get; set; }
        public Review newReview { get; set; }
        public void Run()
        {
            tbName.Text = clientInfo.Name;
            tbSurname.Text = clientInfo.Surname;
            tbTelNo.Text = clientInfo.TelNo;
            foreach(PetTypeInfo pTI in
allPetTypesInfo.listPetTypes)
            {
                dataGridPetTypes.Rows.Add(new string[] {
pTI.KindTitle, pTI.Size, pTI.PricePerDay.ToString(),
pTI.MaxRentDuration.ToString(),
pTI.MinRentDuration.ToString()});
            }
            foreach (Service s in allServices.listServices)
            {
                dataGridServices.Rows.Add(new string[]
{s.KindTitle, s.ServiceName, s.Description,
s.PricePerDay.ToString(), s.Required.ToString()});
            }
            foreach (Pet p in clientPets.listClientPets)
            {
                dataGridClientPets.Rows.Add(new string[] {
p.Name, p.KindTitle, p.Size, p.Weight.ToString() });
                cbClientPet.Items.Add(p);
            }
            foreach (Reservation r in
allClientReservs.listClientReservs)
            {
                dataGridClientReservs.Rows.Add(new string[]
                {
                    r.ReservationNo.ToString(),
                    clientPets.Find(r.PetNo).Name,
                    r.RentStart.ToShortDateString(),
                    r.RentFinish.ToShortDateString(),
                    r.GetSeviceNames(),
```

Лістинг 3.5 – Реалізація інтерфейсу `IViewClientForm` формою `ClientForm`

```
                    r.Price.ToString(),
                    r.GetStringReservStatus()
                });
            }
            cbRating.SelectedIndex = 0;
            foreach (Review r in allReviews.listReviews)
            {
                dataGridAllReviews.Rows.Add(new string[] {
r.Date.ToShortDateString(), r.Rating.ToString(), r.Comment});
            }
            foreach (string kindTitle in
allPetTypesInfo.allKinds)
            {
                cbRgPetKind.Items.Add(kindTitle);
            }
            cbRgPetKind.SelectedIndex = 0;
            foreach (string size in allPetTypesInfo.allSizes)
            {
                cbRgPetSize.Items.Add(size);
            }
            cbRgPetSize.SelectedIndex = 0;
            #region Вкладка "Реєстрація заявки"
            //Початок резервації тільки з наступного дня
            dateStartReserv.MinDate = DateTime.Today.AddDays(1);
            //Кінець резервації мінімум через день після початку
резервації
            dateEndReserv.MinDate = DateTime.Today.AddDays(2);
            #endregion
            Application.Run(this);
        }
        public event EventHandler AskTryToChangeClientInfo;
        public event EventHandler AskAddNewClientPet;
        public event EventHandler AskAddNewReserv;
        public event EventHandler AskFindReservPrice;
        public event EventHandler AskAddNewReview;
        public event EventHandler AskShowReservRooms;
        #endregion
```

Лістинг 3.5, аркуш 2

Форма `ClientForm`, повністю реалізує інтерфейс, повторюючи всі його властивості, події та методи.

У цьому прикладі, метод `Run()` використовується для завантаження даних з полів форми до компонентів прикладного інтерфейсу, таких як `DataGridView` та `ComboBox`. Дані у поля форми завантажуються в свою чергу у представнику, про що буде написано детальніше нижче. Метод викликається також представником, після того як він завершить виклики до

методів моделі та завантажування всіх початкових необхідних даних у властивості форми.

На лістингу 3.6 продемонстрований приклад того, як дані, ведені користувачем зберігаються та передаються представнику для подальшої обробки.

```
private void bAddReview_Click(object sender, EventArgs e)
        {
            if (tbComment.Text == "")
            {
                MessageBox.Show("Напишіть коментар!", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            newReview = new
Review(Convert.ToInt32(cbRating.SelectedItem), tbComment.Text,
DateTime.Now);
            if (AskAddNewReview != null)
                AskAddNewReview(this, EventArgs.Empty);
            dataGridAllReviews.Rows.Clear();
            foreach (Review r in allReviews.listReviews)
            {
                dataGridAllReviews.Rows.Add(new string[] {
r.Date.ToShortDateString(), r.Rating.ToString(), r.Comment });
            }
        }
```

Лістинг 3.6 – Взаємодія представлення та представника на прикладі додавання нового відгуку

Ця функція є обробником події натискання користувачем (в даному випадку клієнтом) на кнопку `bAddReview`. Ціллю цієї функції є додавання нового, написаного клієнтом відгуку у базу даних. Після перевірки на те, що коментар не є порожнім рядком, створюється екземпляр класу `Review`, який був продемонстрований раніше. Цей коментар зберігається у полі `newReview`, яке є поле реалізованого формою інтерфейсу `IViewClientForm`. Далі викликається подія AskAddNewReview, яка є теж елементом реалізованого інтерфейсу. Форма не знає, чи є представники, які підписані на цю подію, чи змінилися якісь дані у властивостях або виникла помилка – формі не потрібно це знати, тому наступним кроком, не залежно від результату

операції додавання відгуку, функція оновлює всі відображувані відгуки. На цьому робота форми і інтерфейсу закінчується.

Тепер розглянемо роботу класу представника. У лістингу 3.7 наведений приклад представника – це є представник для класу форми клієнта `ClientFormPresenter`.

```
public class ClientFormPresenter : AbsShowMessage
    {
        private Model _model;
        private IViewClientForm _viewCF;
        public ClientFormPresenter(IViewClientForm v, Model m)
        {
            _model = m;
            _viewCF = v;
            _viewCF.AskTryToChangeClientInfo += new
EventHandler(OnAskTryToChangeClientInfo);
            _viewCF.AskAddNewClientPet += new
EventHandler(OnAskAddNewClientPet);
            _viewCF.AskAddNewReserv += new
EventHandler(OnAskAddNewReserv);
            _viewCF.AskFindReservPrice += new
EventHandler(OnAskFindReservPrice);
            _viewCF.AskAddNewReview += new
EventHandler(OnAskAddNewReview);
            _viewCF.AskShowReservRooms += new
EventHandler(OnAskShowReservRooms);
        }
        public void Run()
        {
            List<string> buffList = _model.GetClientInfo();
            //Login = "none" - оскільки клієнту не обов'язково
нагадувати його логін
            _viewCF.clientInfo = new UserInfo("none",
buffList[0], buffList[1], buffList[2]);
            _viewCF.allPetTypesInfo =
_model.GetAllPetTypesInfo();
            _viewCF.allServices = _model.GetAllServices();
            _viewCF.clientPets = _model.GetClientPets();
            _viewCF.allClientReservs =
_model.GetClientReservs();
            _viewCF.allReviews = _model.GetAllReviews();
            _viewCF.Run();
        }
private void OnAskAddNewReview(object sender, EventArgs e)
        {
            if (_viewCF.newReview.CheckComment())
            {
```

Лістинг 3.7 – Представника форми клієнта `ClientFormPresenter`

```
                switch (_model.AddNewReview(_viewCF.newReview))
                {
                    case -1:
                        ShowMError("Невідома помилка!");
                        break;
                    case 0:
                        ShowMInfo("Відгук надіслано!");
                        _viewCF.allReviews =
_model.GetAllReviews();
                        break;
                    default:
                        break;
                }
            }
            Else
            {
                MessageBox.Show("Заборонені символи! - )('}{",
"Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }
    }
}
```

Лістинг 3.7, аркуш 2

Представник має у своїх полях посилання на інтерфейс `IViewClientForm`, який в свою чергу реалізований формою `ClientForm`. У своєму конструкторі представник підписує свої функції-обробники подій до подій інтерфейсу, таким чином отримуючи можливість реагувати на події форм, що реалізують цей інтерфейс, та виконувати відповідні до кожної події операції. У методі `Run()`, представник завантажує початкові дані у властивості форми, отримуючи інформацію від методів моделі, які в свою чергу звертаються безпосередньо до бази даних. Також, у тому ж методі, представник викликає реалізований формою метод інтерфейсу `Run()`, який був розглянутий вище.

Тепер розглянемо обробник події додавання нового відгуку `OnAskAddNewReview`. Спочатку він перевіряє, за допомогою метода `CheckComment()` класу `Review` правильність введених даних, потім викликає метод моделі `AddNewReview`, передаючи у якості аргументів поле `newReview` інтерфейсу `_viewCF`, який посилається на конкретну форму

клієнта. Далі обробник, в залежності від результату операції метода моделі, сповіщає користувача про успішну операцію та оновлює дані форми, або сповіщає про помилку та завершую свою роботу.

Останнім кроком у ланцюгу операцій для додавання нового відгуку у застосунку є модель. Приклад методу моделі наведено у лістингу 3.8.

```
public class Model
    {
        NpgsqlConnection conn;
        string userLogin = ""; public string userRole = "";
public int AddNewReview(Review newReview)
        {
            int resultOfOperation = -1;
            using (var cmd = new NpgsqlCommand())
            {
                cmd.Connection = conn;
                cmd.CommandText = $"select * from
AddReview({newReview.Rating}::smallint, '{newReview.Comment}',
'{newReview.Date.ToShortDateString()}'::date)";
                try
                {
                    using (var reader = cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            resultOfOperation =
Convert.ToInt32(reader.GetValue(0).ToString());
                        }
                    }
                }
                catch (NpgsqlException ex)
                {
                    MessageBox.Show($"ОТЛАДОЧНАЯ
ИНФОРМАЦИЯ:\r\nОшибка в классе: {this.ToString()}\r\nТекст
ошибки: {ex.Message}\r\nErrorCode ошибки: {ex.ErrorCode}");
                    resultOfOperation = ex.ErrorCode;
                }
            }
            return resultOfOperation;
        }
    }
}
```

Лістинг 3.8 – Метод `AddNewReview` класу моделі

Цей метод створює екземпляр класу `NpgsqlCommand`, який є компонентом бібліотеки пакетів `Npgsql`, який і зберігає у собі запит, який

буде надісланий до бази даних. Далі метод створює запит у вигляді рядка, який повинен викликати користувацьку функцію `AddReview`, яка реалізована на сервері БД. У якості параметрів функції, метод використовує, переданий до нього від представника екземпляр класу `Review`, відкриті поля якого дозволяють зручно маніпулювати даними про новий відгук клієнта. Після відправлення запиту до БД та отримання відповіді, модель повертає результат операції представнику та завершує свою роботу.

На цьому ланцюг операцій для задачі додавання нового відгуку клієнтом завершено. Усі інші представники та представлення взаємодіють між собою та моделлю за тією ж схемою.

Для підключення до БД використовується екземпляр класу NpgsqlConnection, який також є частиною бібліотеки пакетів Npgsql. Рядок, який використовується класом NpgsqlConnection для підключення до бази даних, виглядає таким чином (див. лістинг 3.9):

```
"Host=localhost;
Username={login};
Password={password};
Database=MyDatabase"
```

Лістинг 3.9 – Команда підключення до бази даних

`Host` позначає посилання на IP адресу бази даних для ІС «Готель для тварин» – у цьому випадку це `localhost`, тобто, БД доступна тільки з комп'ютера, на якому вона встановлена. `Database` позначає ім'я БД, яка в цьому випадку має ім'я `MyDatabase`. `Username` та `Password` позначають відповідно логін та пароль користувача, який намагається підключитися до бази даних.

Підключення до БД реалізується коротким але важливим алгоритмом, тому інформація у роботі [10] є дуже корисною для виконання цієї задачі. Процес підключення до БД починається з того, що користувач вводить логін та пароль, які передаються до функції моделі `TryToLogin` (див. лістинг 3.9). Ця функція вставляє переданий логін та пароль у відповідні місця

командного рядка для підключення до БД та намагається встановити підключення з БД за допомогою функції `Open`. У разі незнайдення переданого логіну у таблиці ролей користувачів у базі даних, або якщо переданий пароль не співпадає з паролем відповідної ролі, то ця функція повертає виключення, яке в свою чергу оброблює функція моделі `TryToLogin`, яка повертає значення 'істинна' у випадку успішного входу, чи значення 'хиба' у випадку виникнення помилки. Надалі цей результат роботи моделі оброблюється представником форми входу та реєстрації, який сповіщає клієнта про виникнення помилки, або повертає управління основному класу контролера, сповіщаючи його про успішне встановлення з'єднання з БД. У свою чергу контролер, в залежності від групи до якої відноситься користувач, запускає відповідний представник форми. Група, до якої відноситься користувач, визначається у раніше зазначеної функції `TryToLogin` (див. лістинг 3.10).

```
public bool TryToLogin(string login, string password)
        {
            string connectionString =
$"Host=localhost;Username={login};Password={password};Database=M
yDatabase";
            //string connectionString =
"Host=localhost;Username=postgres;Password=postgres;Database=MyD
atabase";
            // Створення підключення
            conn = new NpgsqlConnection(connectionString);
            try
            {
                conn.Open();
                userLogin = conn.UserName;
                //Знаходження групової ролі користувача:
                using (var cmd = new NpgsqlCommand())
                {
                    cmd.Connection = conn;
                    cmd.CommandText = $"SELECT rolname FROM
pg_roles WHERE oid IN (SELECT roleid FROM pg_auth_members WHERE
member IN (SELECT oid FROM pg_roles WHERE rolname =
'{userLogin}'));";
                    using (var reader = cmd.ExecuteReader())
                    {
                        while (reader.Read())
```

Лістинг 3.10 – Функція моделі `TryToLogin`

```
                        {
                            userRole =
reader.GetValue(0).ToString();
                        }
                    }
                }
            }
            catch (NpgsqlException ex)
            {
                conn.Close();
                return false;
            }
            // Якщо акаунт у співробітника є, але доступу до
ролі немає
            if(userRole == "")
            {
                return false;
            }
            return true;
        }
```

Лістинг 3.10, аркуш 2

Для реалізації системи відео спостереження використовується бібліотека LibVLCSharp. Вона потрібна для відображення RTSP трансляцій. Для відображення YouTube трансляцій використовується компонент WinForms під назвою WebView2. Спочатку треба ініціалізувати об’єкт бібліотеки LibVLCSharp та підписати необхідні обробники подій (див. лістинг 3.11).

```
Core.Initialize(); // Ініціалізація LibVLC
libVLC = new LibVLC();
mediaPlayer = new MediaPlayer(libVLC);
videoView1.MediaPlayer = mediaPlayer;
mediaPlayer.EncounteredError += MediaPlayer_EncounteredError;
mediaPlayer.Playing += MediaPlayer_Playing;
```

Лістинг 3.11 – Ініціалізація змінних бібліотеки LibVLCSharp

Основний код для відображення відео розміщений у функції обробника події зміни обраної кімнати або тварини, в залежності від того, у формі якого користувача реалізується функція трансляції відео (клієнт, менеджер або ветеринар) Для прикладу наведено реалізація механізму для менеджера

(див. лістинг 3.12). Якщо в обраній кімнаті камера з RTSP трансляцією, то компонент WebView2 деактивується та робиться невидимим, а компонент videoView1, який використовується бібліотекою LibVLCSharp, активізується та починає працювати з відео.

```
private void dataGridPetRooms_SelectionChanged(object sender,
EventArgs e)
{
    if (dataGridPetRooms.SelectedRows.Count == 0) return;
    if (dataGridPetRooms.SelectedRows[0].Tag == null) return;
    (int roomNo, string streamURL, string streamType) bufferTag
= ((int, string, string))dataGridPetRooms.SelectedRows[0].Tag;
    tbPetRoomStreamURL.Text = bufferTag.streamURL;
    cbPetRoomStreamType.SelectedItem = bufferTag.streamType;
    cbPetKindRoom.SelectedItem =
dataGridPetRooms.SelectedRows[0].Cells[1].Value;
    cbPetSizeRoom.SelectedItem =
dataGridPetRooms.SelectedRows[0].Cells[2].Value;
    #region Код для трансляции видео
    if (bufferTag.streamURL == "")
    {
        groupBoxForVideo.Text =
groupBoxForVideo.Text.Substring(0,
groupBoxForVideo.Text.IndexOf(':') + 1)
            + " У цієї кімнати немає камери.";
        StopWebViewVideo();
        StopVLCVideo();
        chbifPetRoomHasCamera.Checked = false;
        return;
    }
    chbifPetRoomHasCamera.Checked = true;
    groupBoxForVideo.Text = groupBoxForVideo.Text.Substring(0,
groupBoxForVideo.Text.IndexOf(':') + 1)
            + "";
    switch (bufferTag.streamType)
    {
        case "YouTube":
            StopVLCVideo();
            webView21.Visible = true;
            try
            {
                webView21.Source = new Uri(bufferTag.streamURL);
            }
            catch (UriFormatException)
            {
                webView21.Source = new Uri("about:blank");
                MessageBox.Show("Помилка: Неправильний URL!",
"Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

Лістинг 3.12 – Функція відображення відео

```
            }
            break;
        case "RTSP":
            StopWebViewVideo();
            videoView1.Visible = true;
            groupBoxForVideo.Text =
groupBoxForVideo.Text.Substring(0,
groupBoxForVideo.Text.IndexOf(':') + 1)
            + " Відео завантажується...";
            this.Enabled = false;
            mediaPlayer.Play(new Media(libVLC,
bufferTag.streamURL, FromType.FromLocation));
            break;
        default: break;
    }
    #endregion
}
```

Лістинг 3.12, аркуш 2

Для оброблення помилок та коректної роботи відео плеєра бібліотеки LibVLCSharp також реалізовані функції обробники подій та допоміжні функції для більш зручного зупинення трансляції (див. лістинг 3.13).

```
private async Task StopWebViewVideo()
{
    await webView21.EnsureCoreWebView2Async(null);

webView21.CoreWebView2.NavigateToString($"<html><body></body></h
tml>");
    webView21.Visible = false;
}
private void StopVLCVideo()
{
    mediaPlayer.Stop();
    videoView1.Visible = false;
}
private void MediaPlayer_EncounteredError(object sender,
EventArgs e)
{
    //Invoke потрібен, оскільки VLC намагається запустити відео
у фоновому режимі, а нам потрібно зробити this.Enabled = true
    // для цього потрібно звертатися в головний потік, для цього
і потрібен Invoke
    this.Invoke((Action)(() =>
    {
        MessageBox.Show("При завантаженні відео виникла
помилка", "Помилка", MessageBoxButtons.OK,
```

Лістинг 3.13 – Допоміжні функції для системи відео трансляцій

```
MessageBoxIcon.Error);
        this.Enabled = true;
    }));
}
private void MediaPlayer_Playing(object sender, EventArgs e)
{
    this.Invoke((Action)(() =>
    {
        this.Enabled = true;
    }));
}
```

Лістинг 3.13, аркуш 2

3.5 Мобільний застосунок

Мобільний застосунок написаний у спрощеній версії шаблону програмування MVP. Враховуючи обмеження та набір функцій, які були описані у розділі 2.4, мобільний застосунок має всього 3 пари структурних компонентів: файл формату xml, який визначає зовнішній вигляд сторінки та налаштування компонентів користувацького інтерфейсу та файл класу сторінки, де визначені обробники подій та методи взаємодії з класом моделі.

Сторінки авторизації, клієнта та ветеринара є аналогічними за структурою, тому для прикладу розглянуто реалізацію сторінки ветеринара.

Спочатку визначається зовнішній вигляд сторінки та компонентів у xml файлі (див. лістинг 3.14). В цьому прикладі, на сторінці є елемент `VerticalStackLayout`, який грає роль компонента групування елементів сторінки, а саме розташування елементів у одному стовпці зверху вниз. Далі йде компонент `Label`, який слугує для відображення інформації для користувача. Наступний компонент `Picker` потрібен для того, аби ветеринар мав можливість обирати тварин для перегляду трансляцій з їх кімнат. Серед параметрів цього компоненту визначений параметр `x:Name` та `SelectedIndexChanged` – перший потрібен для того, аби мати можливість звертатися до компонента з файлу класу сторінки; другий потрібен для призначення конкретного обробника подій для події зміни індексу обраного

елементу (тобто, обраного вихованця). Останнім елементом є `WebView`, який упакований у рамку для більш наочного відображення. Цей елемент також має визначений параметр `x:Name`, адже саме в цьому компоненті будуть відображатися трансляції з кімнат тварин. Інші вказані параметри компонентів призначені для налаштування коректного відображення компонента.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
          xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="DiplomaMobileApp.VetPage"
             Title="VetPage">
    <VerticalStackLayout
        Padding="20,0"
        Spacing="10">
        <Label
            Text="Оберіть вихованця:"
            VerticalOptions="Center"
            HorizontalOptions="Center" />
        <Picker
            x:Name="pickerPets"
            BackgroundColor="AliceBlue"
SelectedIndexChanged="pickerPets_SelectedIndexChanged">
        </Picker>
        <Border Stroke="Black" StrokeThickness="1">
            <WebView
                x:Name="webView"
                HeightRequest="500"
                WidthRequest="400"/>
        </Border>
    </VerticalStackLayout>
</ContentPage>
```

Лістинг 3.14 – Xml файл сторінки ветеринара

Далі визначається сам клас сторінки, у якому описані функції взаємодії з класом моделі та функції обробників подій (див. лістинг 3.15). В цьому прикладі, спочатку визначаються поля класу: `allPets` – список інформації про всіх тварин, які потребують медичного догляду на поточний момент (крім тих, які призначені на інших ветеринарів); `items` – список, у якому безпосередньо зберігаються елементи, які будуть додані до компонента

Picker, він потрібен для приховання зайвої інформації про тварину від користувача та для більш зручного відображення інформації. Далі визначається конструктор сторінки, у якому програма безпосередньо звертається до класу моделі з запитом, який повинен повернути інформацію про тварин і трансляції у їхніх кімнатах, яка потім у більш зручному вигляді додається до компонента Picker. Далі визначається функція обробника події зміни обраної ветеринаром тварини – функція pickerPets_SelectedIndexChanged. Функція перевіряє, чи знаходиться тварина у готелі на поточний момент, чи є у кімнаті камера та який тип трансляції використовує ця камера. Якщо всі перевірки пройдені, функція, за допомогою компонента webView виводить трансляцію на екран.

```
namespace DiplomaMobileApp;
public partial class VetPage : ContentPage
{
    public List<(
        string petName,
        string petKind,
        string petType,
        string petRoomNo,
        string streamType,
        string streamURL
        )> allPets { get; set; }
    = new List<(string petName, string petKind, string petType,
string petRoomNo, string streamType, string streamURL)>();
    private List<string> items = new List<string>();
    public VetPage()
    {
        InitializeComponent();
        List<List<string>> buffList = Model.SelectQuery("select
* from AllReservs_Vet ");
        foreach (List<string> list in buffList)
        {
            allPets.Add((list[2], list[4], list[5], list[8],
list[13], list[12]));
            items.Add($"{list[2]}, {list[5]} {list[4]}, Номер
кімнати: {list[8]}");
        }
        pickerPets.ItemsSource = null;
        pickerPets.ItemsSource = items;
    }
```

Лістинг 3.15 – Код класу VetPage сторінки ветеринара

```
    private void pickerPets_SelectedIndexChanged(object sender,
EventArgs e)
    {
        int selectedPetIndex = pickerPets.SelectedIndex;
        if (allPets[selectedPetIndex].petRoomNo == "")
        {
            DisplayAlert("Увага", "Цей вихованець не має
підтверджених резервацій на поточний час.", "Ок");
            webView.Source = null;
            return;
        }
        if (allPets[selectedPetIndex].streamURL == "")
        {
            DisplayAlert("Увага", "У кімнаті цього вихованця
немає камери.", "Ок");
            webView.Source = null;
            return;
        }
        if (allPets[selectedPetIndex].streamType == "RTSP")
        {
            DisplayAlert("Увага", "RTSP трансляції не
підтримуються в мобільному додатку", "Ок");
            webView.Source = null;
            return;
        }
        webView.Source = allPets[selectedPetIndex].streamURL;
    }
}
```

Лістинг 3.15, аркуш 2

Взаємодія застосунку з сервером бази даних здійснюється за тим самим механізмом, що й у десктопному застосунку (див. розділ 3.3).

3.6 Безпека інформаційної системи

Безпека на рівні БД забезпечується шляхом створення ролей і наділення їх відповідними привілеями. В ІС «Готель для тварин» реалізовано шість ролей: клієнт, асистент, грумер, завскладом, менеджер та ветеринар.

У таблиці Ж.1 додатка Ж наведені привілеї ролей на таблиці БД. При цьому використані такі позначення: C (Create) – створення (додавання) даних, R (Read) – читання даних, U (Update) – оновлення даних, D (Delete) – видалення даних, E (Execute) – виконання процедури (функції).

Запити на створення ролей і наділення їх відповідними привілеями наведені у додатку И.

Паролі користувачів зберігаються у зашифрованому виді у системній таблиці `pg_roles` на сервері бази даних. Паролі автоматично шифруються при виконанні команди створення ролі, наприклад `create role [логін] with login password ['пароль']`.

Визначення ролі користувача виконується за допомогою такого запиту (лістинг 3.16).

```
SELECT rolname
FROM pg_roles
WHERE oid IN (
    SELECT roleid
    FROM pg_auth_members
    WHERE member IN (
        SELECT oid
        FROM pg_roles
        WHERE rolname = '[логін_користувача]'
    )
);
```

Лістинг 3.16 – Запит для визначення ролі користувача

Виконуючи запити до таблиць `pg_roles`, яка містить усі ролі БД та `pg_auth_members`, яка містить групи, до яких належать ролі, та знаходячи роль з відповідним логіном, який передав користувач, знаходиться група, до якої належить поточний користувач.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Інтерфейс авторизації та реєстрації

При запуску програми користувач бачить вікно (рис. 4.1), що містить поля для воду даних для авторизації. Також форма містить вікно реєстрації клієнта, де користувач може вписати свої персональні дані, логін, пароль та натиснути кнопку «Зареєструватись», після чого, відкриється інтерфейс клієнта. Якщо користувач вже має свій обліковий запис, то, ввівши коректні дані у вікні авторизації, відкриється та форма, в залежності від групи, до якої належить користувач. У мобільному застосунку, реалізований тільки механізм авторизації для ветеринарів і клієнтів (рис. 4.2).

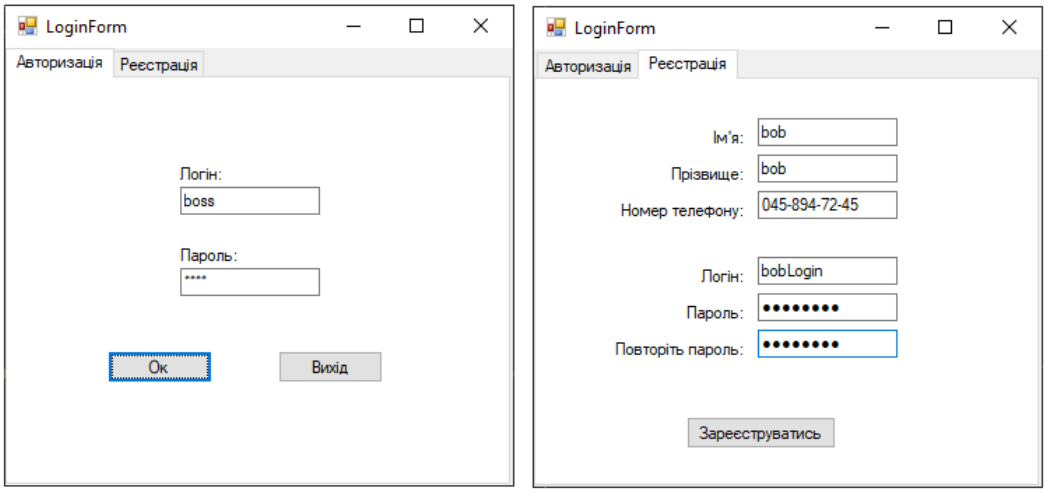


Рисунок 4.1 – Форма авторизації та реєстрації користувача

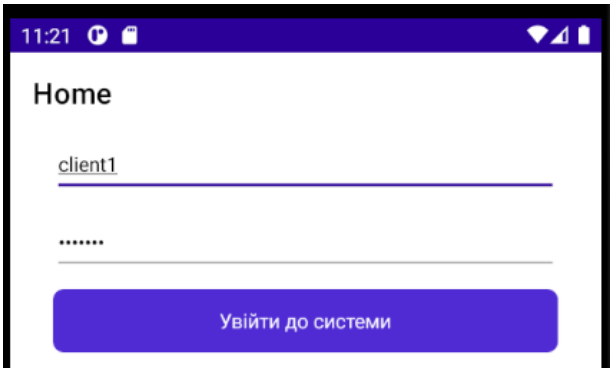


Рисунок 4.2 – Форма авторизації у мобільному застосунку

4.2 Інтерфейс клієнта

На рисунку 4.3 продемонстровано вікно «Мій профіль» форми клієнта, яке є початковим вікном, на яке потрапляє клієнт, після авторизації.

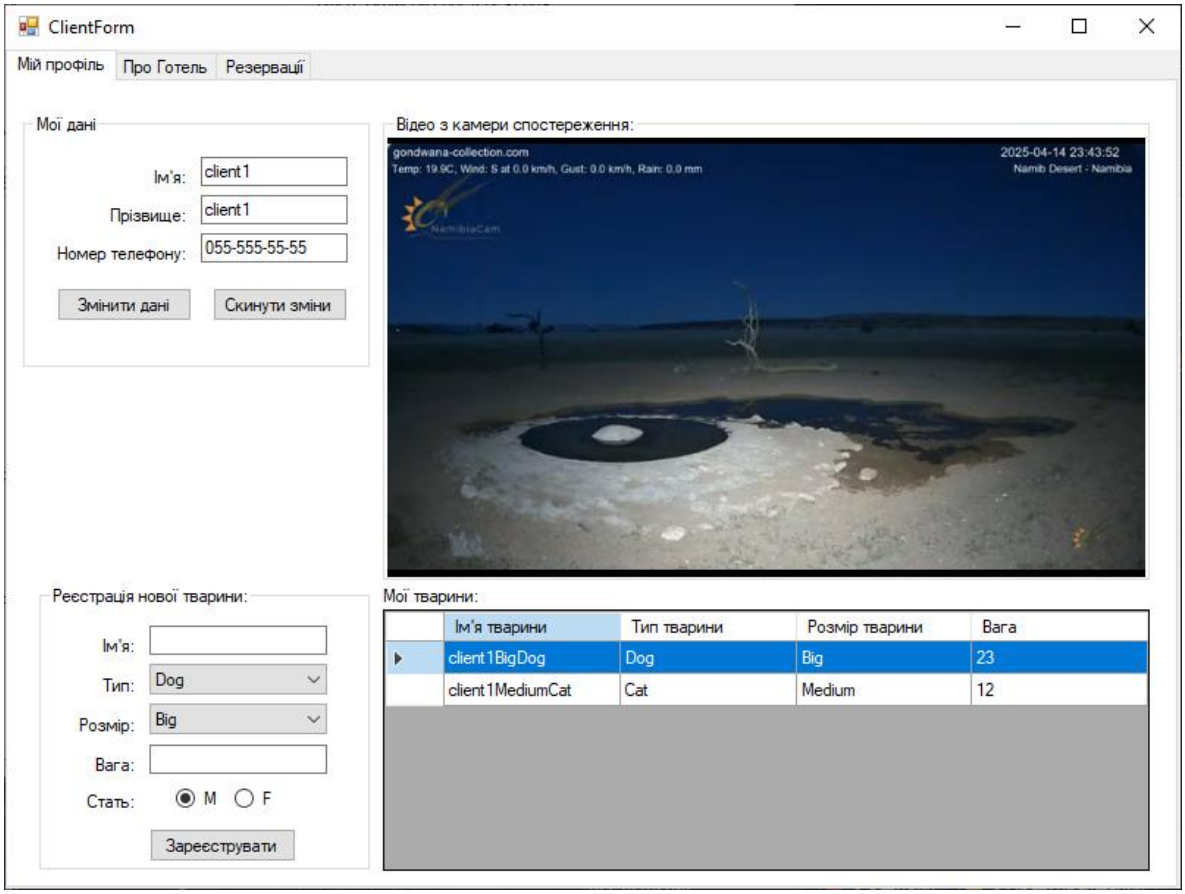


Рисунок 4.3 – Вікно «Мій профіль» форми клієнта

На цьому вікні клієнт може виконувати такі дії:

– переглядати та змінювати свої персональні дані у області вікна «мої дані»;

– реєструвати нових вихованців у області вікна «реєстрація нової тварини»;

– переглядати інформацію про своїх вихованців у області вікна «мої тварини»;

– дивитися трансляції з камер у кімнатах тих тварин, які мають підтверджені резервації на поточний час.

У мобільному застосунку, клієнти мають можливість дивитися трансляцію з камери обраного вихованця (рис. 4.4).

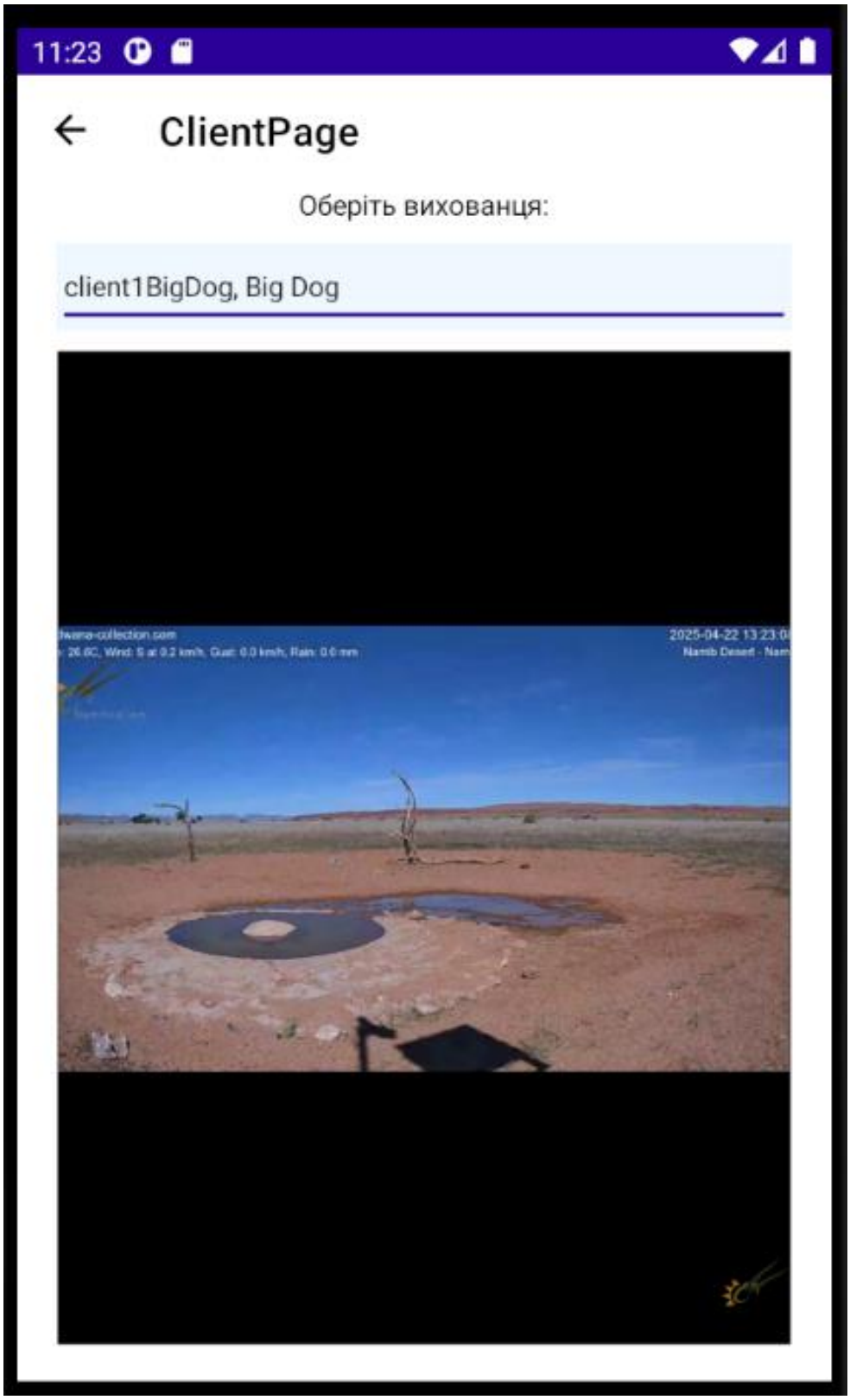


Рисунок 4.4 – Форми клієнта у мобільному застосунку

На рисунку 4.5 продемонстровано вікно «Про Готель» форми клієнта.

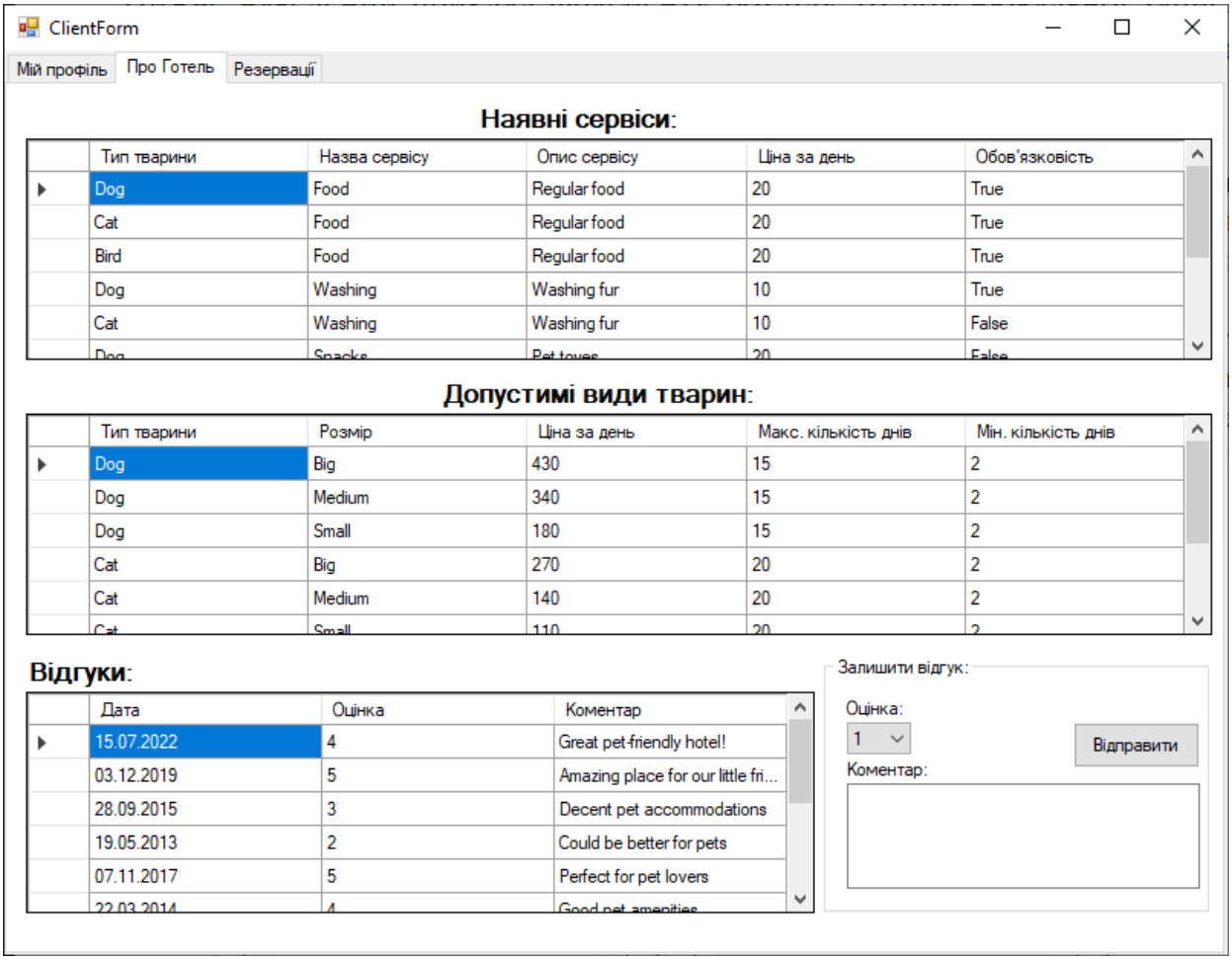


Рисунок 4.5 – Вікно «Про Готель» форми клієнта

На цьому вікні клієнт може виконувати такі дії:

– переглядати інформацію про наявні сервіси у таблиці «наявні сервіси»;

– переглядати інформацію про допустимих тварин у таблиці «допустимі види тварин»;

– переглядати відгуки інших клієнтів у таблиці «відгуки»;

– залишити свій відгук у області вікна «залишити відгук», написавши коментар та залишивши оцінку від 1 до 5.

На рисунку 4.6 продемонстровано вікно «Резервації» форми клієнта.

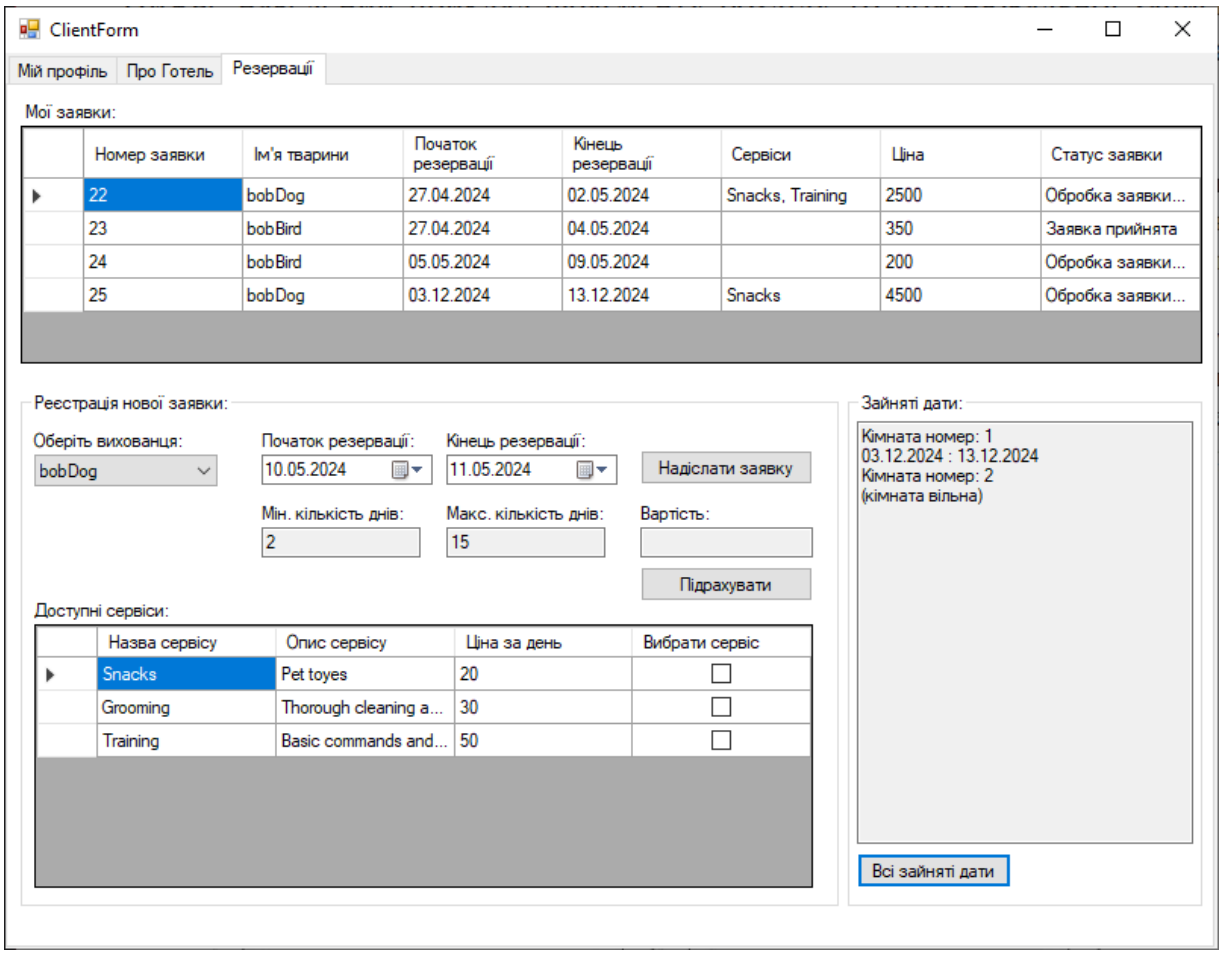


Рисунок 4.6 – Вікно «Резервації» форми клієнта

На цьому вікні клієнт може виконувати такі дії:

– переглядати інформацію про свої резервації у таблиці «мої заявки»;

– додавати нову заявку у області вікна «реєстрація нової заявки», вказуючи вихованця, початок та кінець резервації та, за бажанням, обравши додаткові сервіси;

– переглядати кімнати та періоди, коли вони є зайнятими у області вікна «зайняті дати»;

– обчислювати ціну резервації, на підставі обраного вихованця, додаткові сервісів та тривалості резервації.

4.3 Інтерфейс асистента

На рисунку 4.7 продемонстровано вікно «Мій профіль» форми асистента, яке є початковим вікном для асистента, після авторизації.

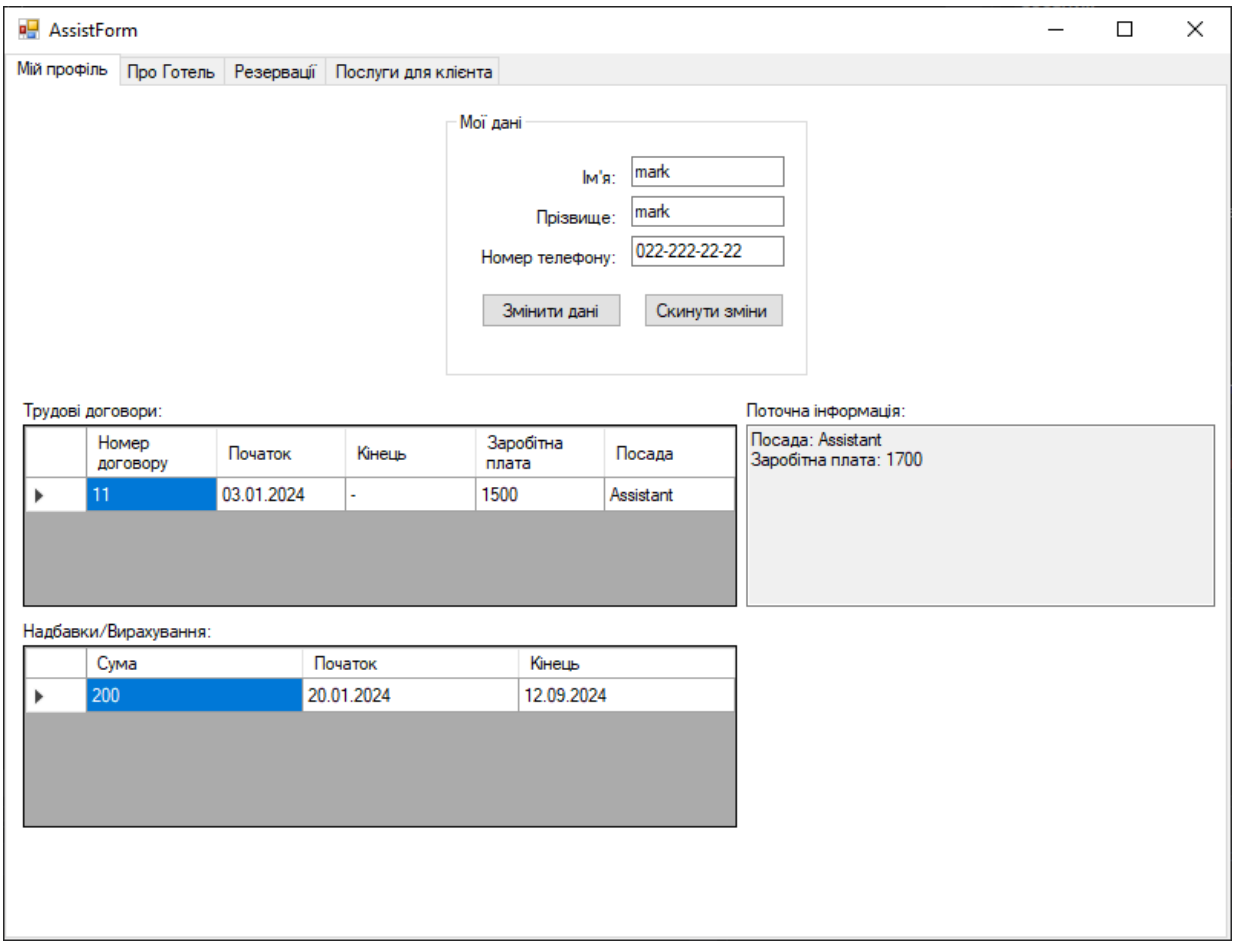


Рисунок 4.7 – Вікно «Мій профіль» форми асистента

На цьому вікні асистент може виконувати такі дії:

– переглядати та змінювати свої персональні дані у області вікна «мої дані»;

– переглядати свої трудові договори у таблиці «трудові договори»;

– переглядати свої надбавки/вирахування у таблиці «надбавки/вирахування»;

– переглядати свою поточну робочу інформацію у області вікна «поточна інформація».

На рисунку 4.8 продемонстровано вікно «Про Готель» форми асистента.

На цьому вікні асистент може виконувати такі дії:

– переглядати інформацію про наявні сервіси у таблиці «наявні сервіси»;

– переглядати інформацію про допустимих тварин у таблиці «допустимі види тварин».

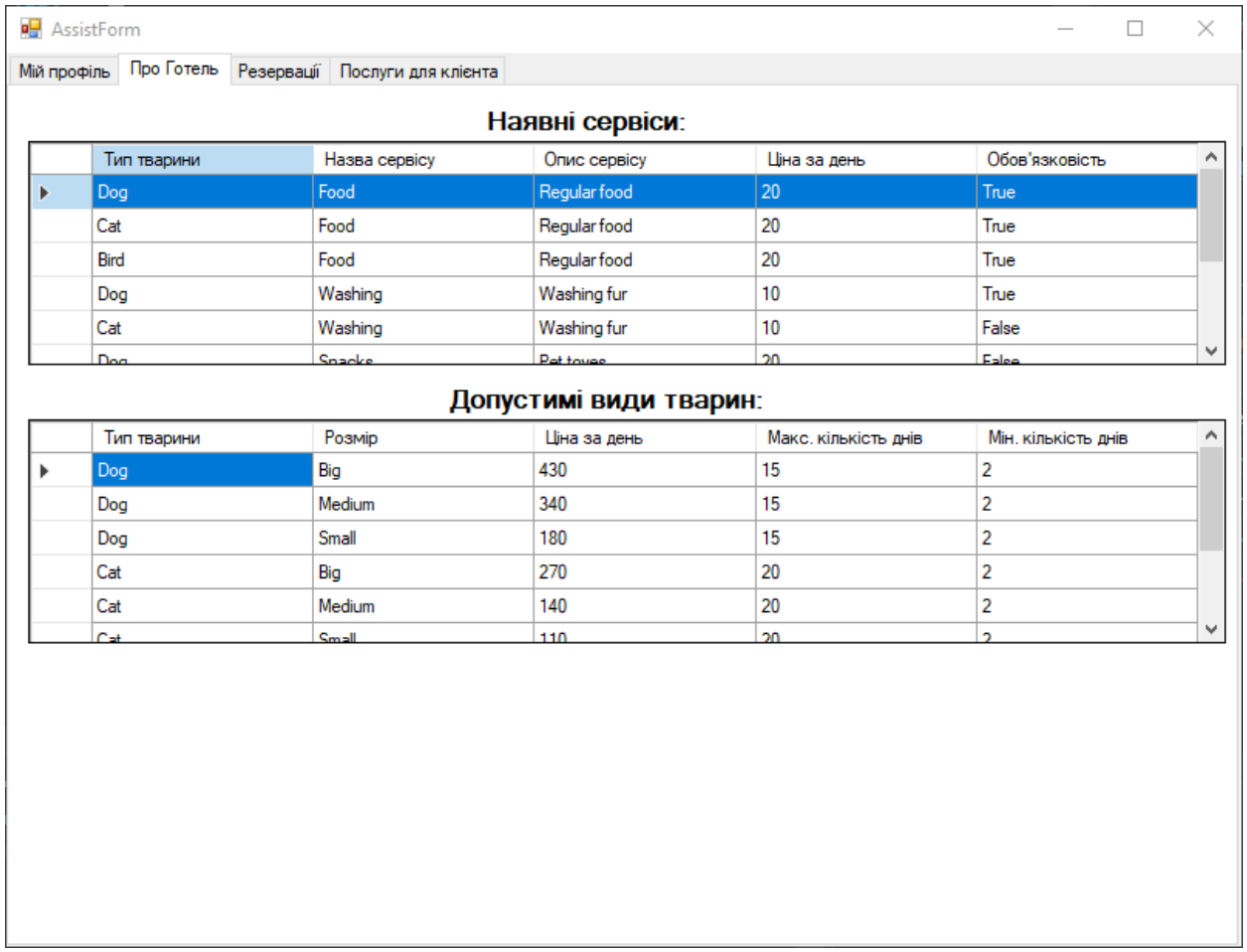


Рисунок 4.8 – Вікно «Про Готель» форми асистента

На рисунку 4.9 продемонстровано вікно «Резервації» форми асистента.

На цьому вікні асистент може виконувати такі дії:

– переглядати інформацію про заявки від клієнтів за такими фільтрами: всі/прийняті/у стані очікування/сьогодні завершуються;

– переглядати інформацію про клієнта обраної в таблиці резервації;

– підтвердити або видалити обрану заявку, натиснувши на кнопки з відповідними назвами.

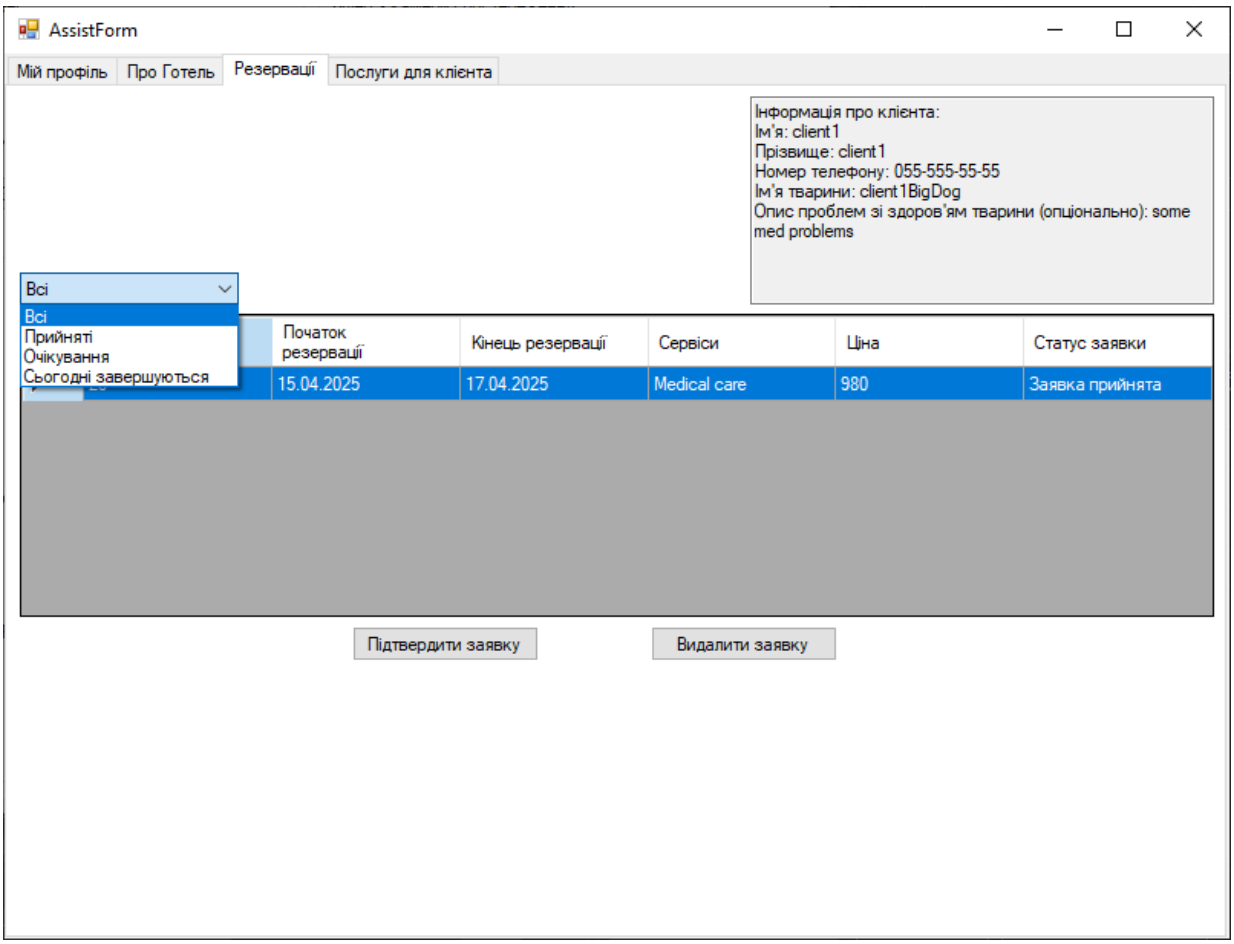


Рисунок 4.9 – Вікно «Резервації» форми асистента

На рисунку 4.10 продемонстровано вікно «Послуги для клієнта» форми асистента.

На цьому вікні асистент може виконувати такі дії:

– переглядати інформацію про клієнта, ввівши його логін у області вікна «знайти клієнта»;

– реєстрація нових вихованців для обраного клієнта у області вікна «реєстрація нової тварини»;

– реєстрація нових клієнтів у області вікна «реєстрація клієнта»;

– додавати нову заявку у області вікна «реєстрація нової заявки», вказуючи вихованця, початок та кінець резервації та, за бажанням, обравши додаткові сервіси для обраного клієнта;

– переглядати кімнати та періоди, коли вони є зайнятими у області вікна «зайняті дати»;

– обчислювати ціну резервації, на підставі обраного вихованця, додаткові сервісів та тривалості резервації.

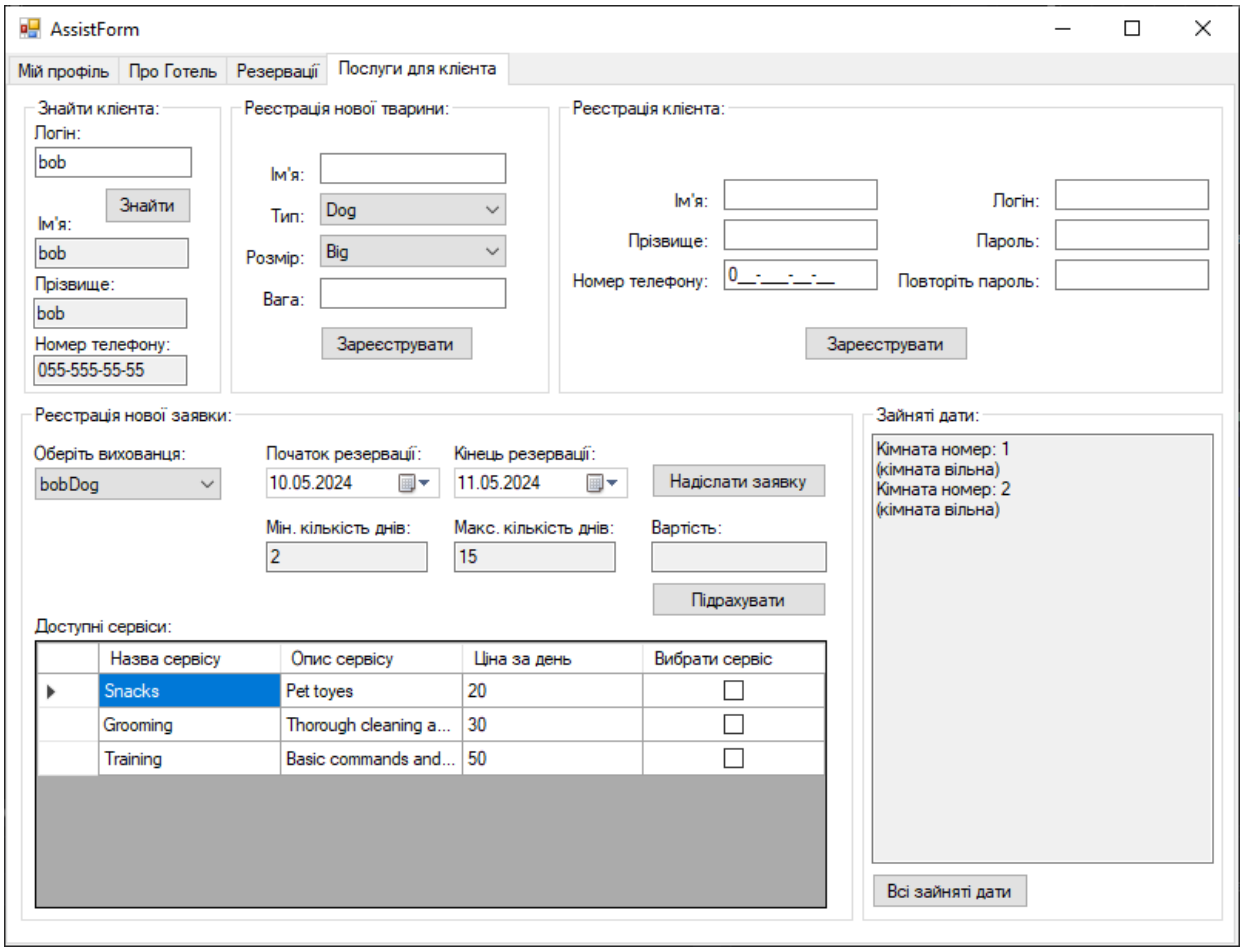


Рисунок 4.10 – Вікно «Послуги для клієнта» форми асистента

4.4 Інтерфейс грумера

На рисунку 4.11 продемонстровано вікно «Мій профіль» форми грумера, яке є початковим вікном, на яке потрапляє грумер, після авторизації.

На цьому вікні грумер може виконувати такі дії:

– переглядати та змінювати свої персональні дані у області вікна «мої дані»;

– переглядати свої трудові договори у таблиці «трудові договори»;

– переглядати свої надбавки/вирахування у таблиці «надбавки/вирахування»;

– переглядати свою поточну робочу інформацію у області вікна «поточна інформація».

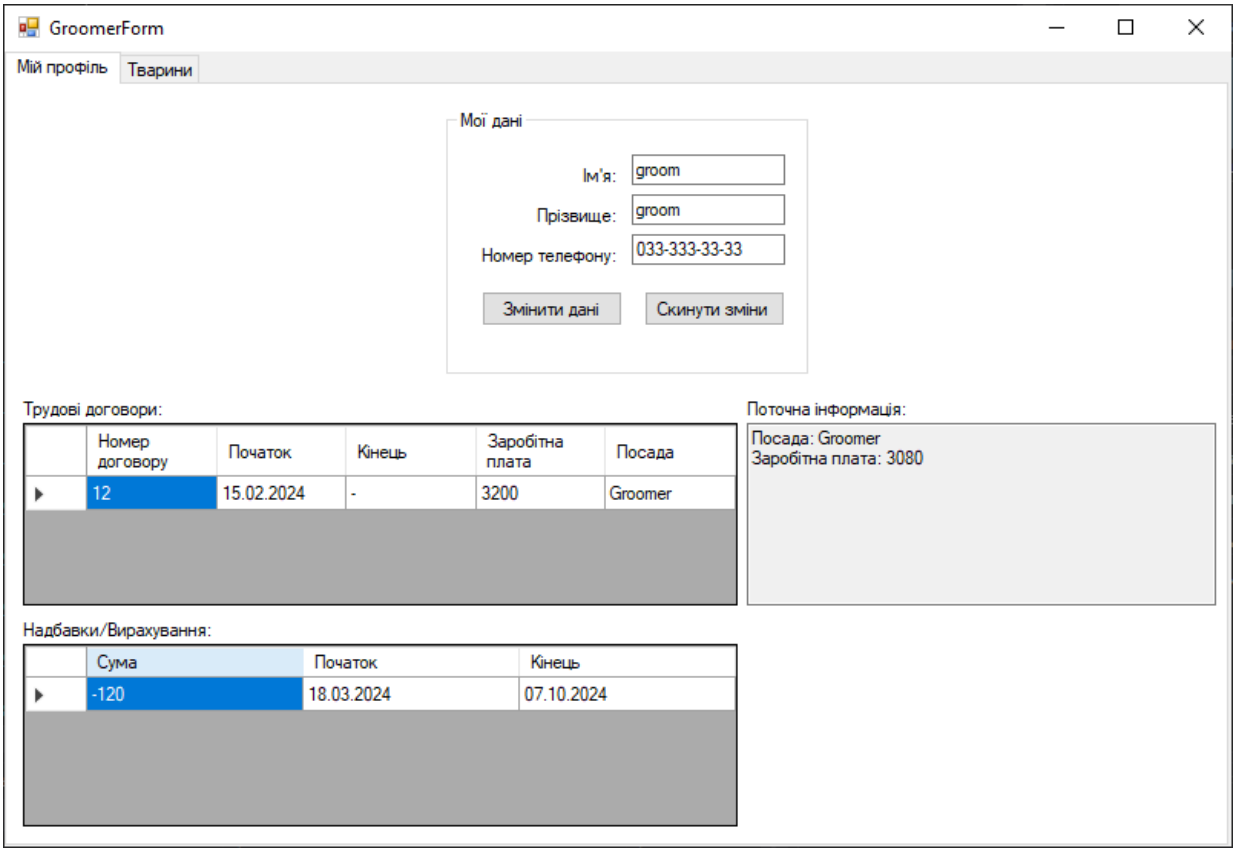


Рисунок 4.11 – Вікно «Мій профіль» форми грумера

На рисунку 4.12 продемонстровано вікно «Тварини» форми грумера.

На цьому вікні грумер може виконувати такі дії:

– переглядати тварин у таблиці «тварини для обслуговування сьогодні», які у поточний час знаходяться у готелі та потребують догляду;

– переглядати інформацію про ресурси, які потребують сервіси, які підключені до обраної резервації у таблиці «підключені сервіси».

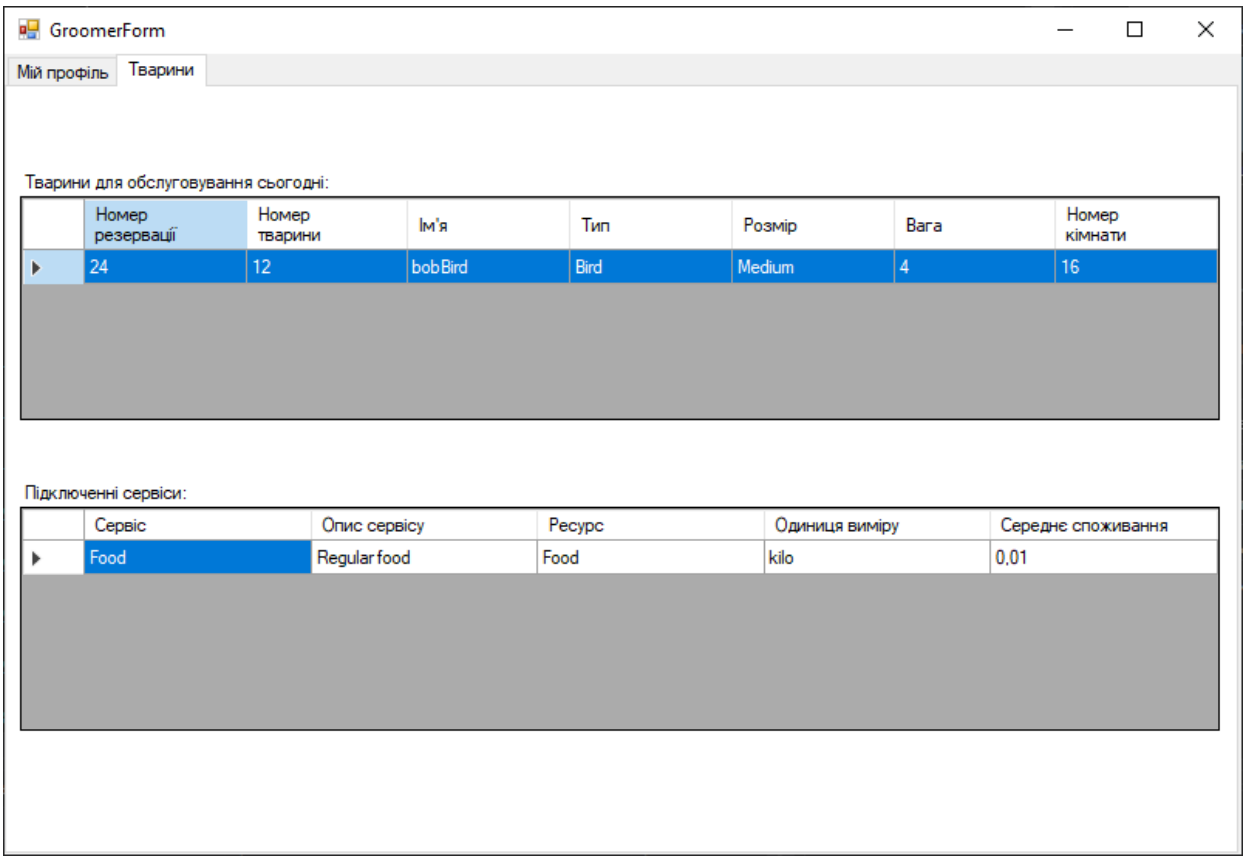


Рисунок 4.12 – Вікно «Тварини» форми грумера

4.5 Інтерфейс завідувача складу

На рисунку 4.13 продемонстровано вікно «Мій профіль» форми завскладом, яке є початковим вікном, на яке потрапляє завскладом, після авторизації.

На цьому вікні завскладом може виконувати такі дії:

– переглядати та змінювати свої персональні дані у області вікна «мої дані»;

– переглядати свої трудові договори у таблиці «трудові договори»;

– переглядати свої надбавки/вирахування у таблиці «надбавки/вирахування»;

переглядати свою поточну робочу інформацію у області вікна «поточна інформація». На рисунку 4.14 продемонстровано вікно «Звіти про ресурси» форми завсклада.

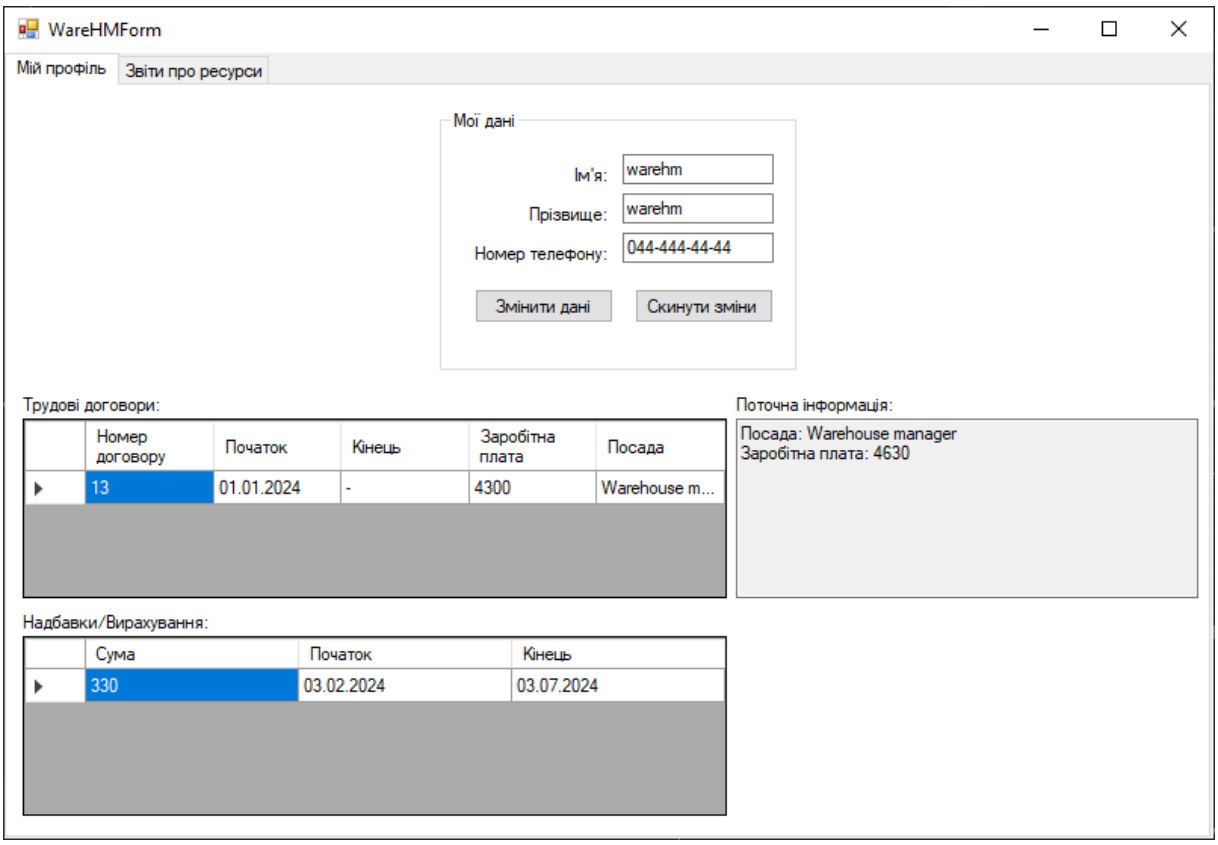


Рисунок 4.13 – Вікно «Мій профіль» форми завсклада

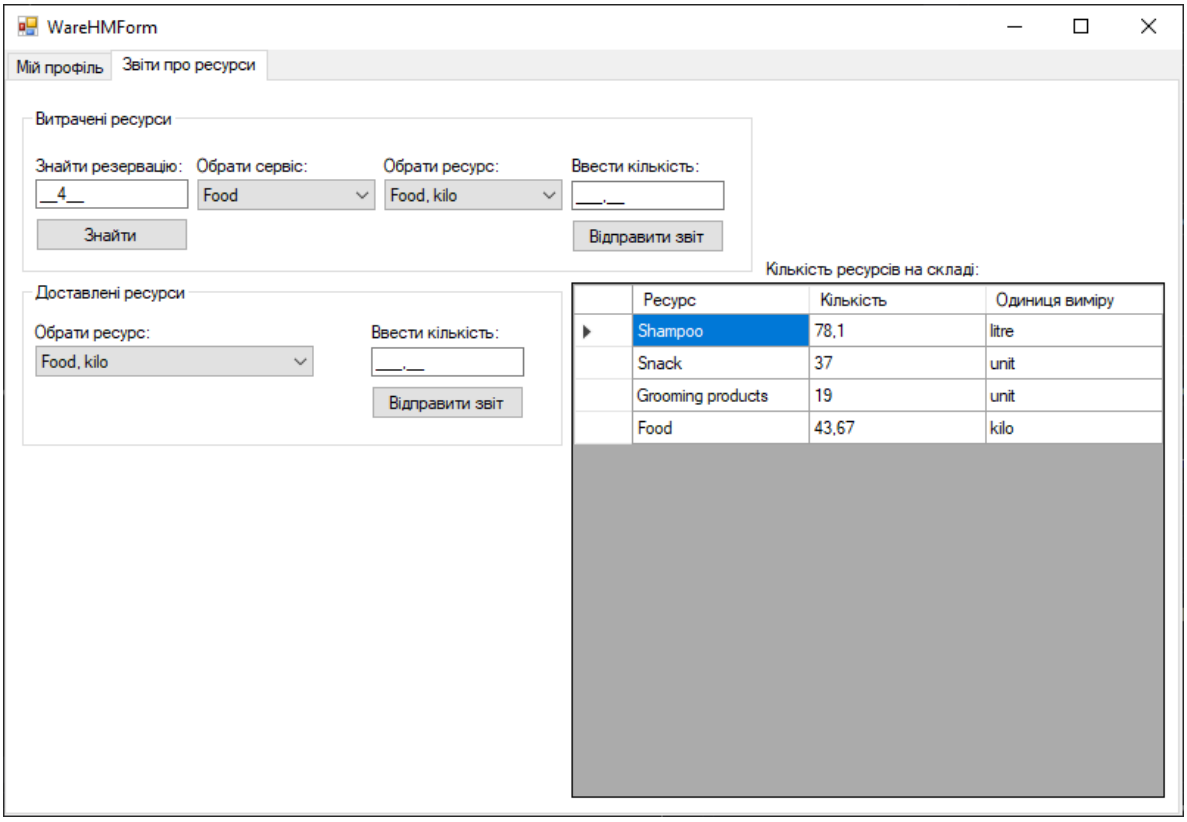


Рисунок 4.14 – Вікно «Звіти про ресурси» форми завсклада

На цьому вікні завскладом може виконувати такі дії:

– переглядати кількість ресурсів, які залишились на складі, у таблиці «кількість ресурсів на складі»;

– додавати звіт про витрачені ресурси у області вікна «витрачені ресурси», вказуючи ресурс та кількість ресурсу, що було витрачено, та резервацію і сервіс, на який витрачено обраний ресурс;

– додавати звіт про доставлені ресурси у області вікна «доставлені ресурси», вказуючи ресурс, який було доставлено та його кількість.

4.6 Інтерфейс менеджера

На рисунку 4.15 продемонстровано вікно «Про Готель» форми менеджера, яке є початковим вікном, на яке потрапляє менеджер, після авторизації.

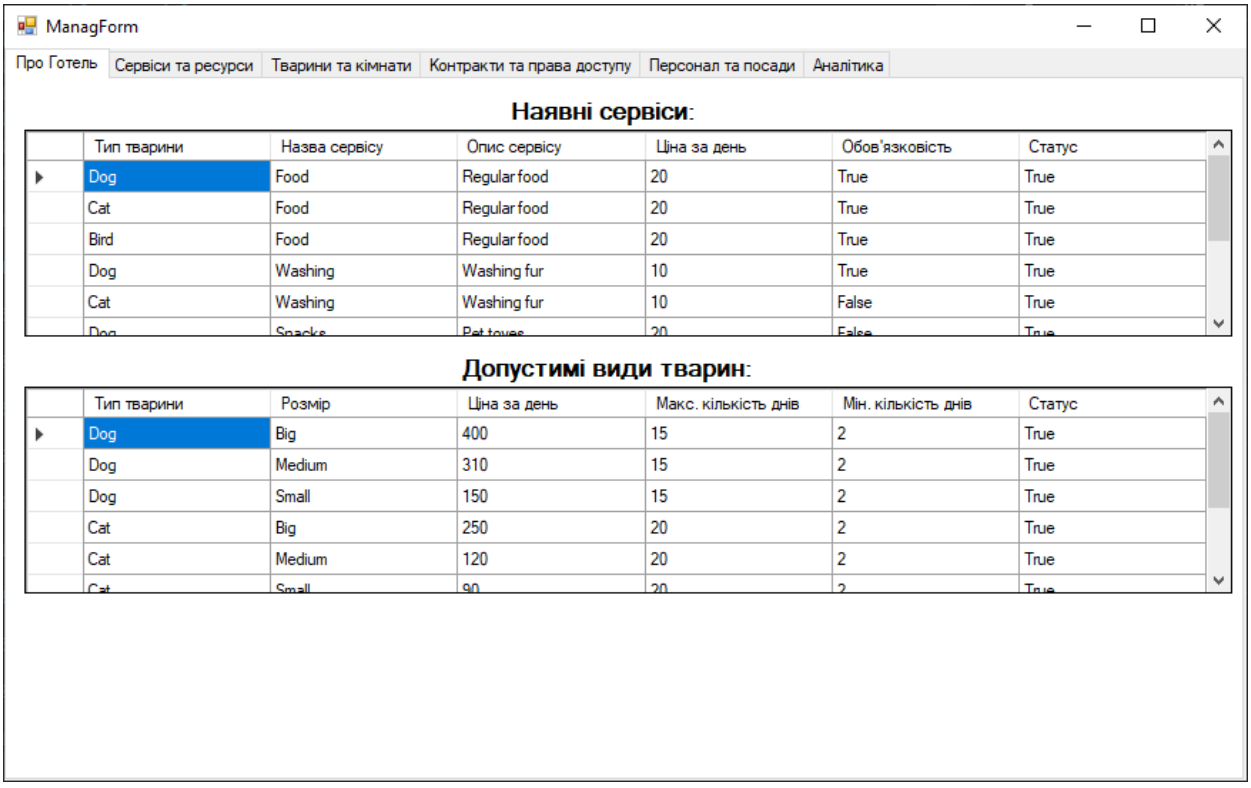


Рисунок 4.15 – Вікно «Про Готель» форми менеджера

На цьому вікні менеджер може виконувати такі дії:

– переглядати інформацію про наявні сервіси у таблиці «наявні сервіси»;

– переглядати інформацію про допустимих тварин у таблиці «допустимі види тварин».

На рисунку 4.16 продемонстровано вікно «Сервіси та ресурси» форми менеджера.

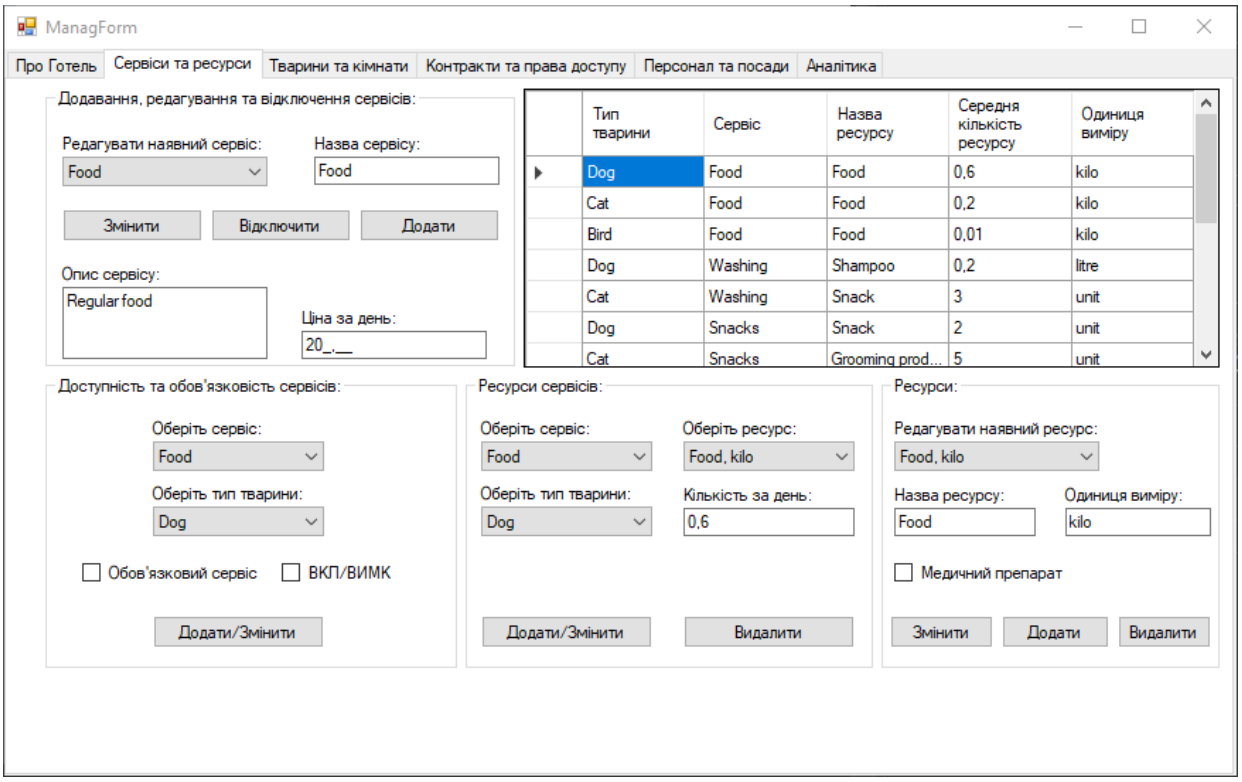


Рисунок 4.16 – Вікно «Сервіси та ресурси» форми менеджера

На цьому вікні менеджер може виконувати такі дії:

– переглядати інформацію у таблиці справа про тип та кількість ресурсу, який в середньому витрачається на відповідний сервіс, для відповідного типу тварин;

– додавати, виключати та редагувати інформацію про сервіси у області вікна «додавання, редагування та відключення сервісів»;

– додавати та віднімати можливість використання обраного сервіса для обраного типа тварин та змінювати його обов’язковість у області вікна «доступність та обов’язковість сервісів»;

– додавати, видаляти та редагувати кількість та тип ресурсу, який використовується обраним сервісом для обраного типу тварин у області вікна «ресурси сервісів»;

– додавати, видаляти та редагувати тип ресурсу, його одиницю виміру та помічати його як медичний препарат у області вікна «ресурси».

На рисунку 4.17 продемонстровано вікно «Тварини та кімнати» форми менеджера.

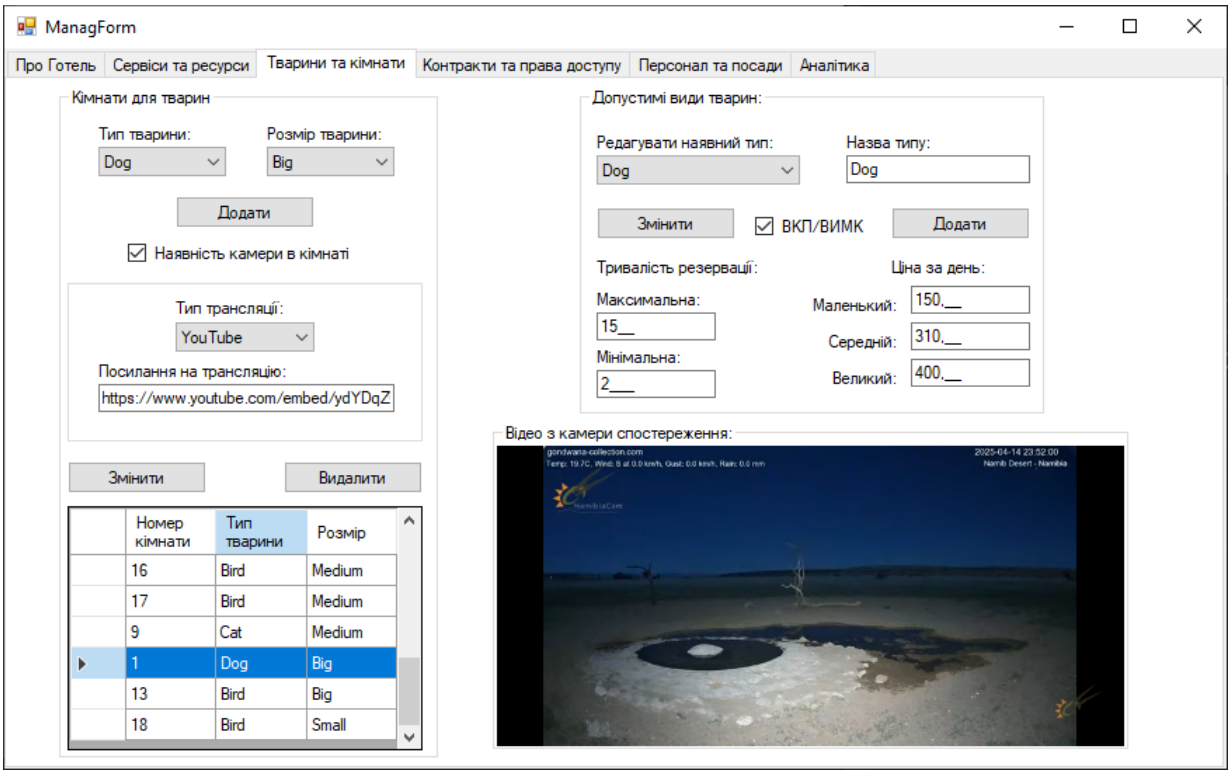


Рисунок 4.17 – Вікно «Тварини та кімнати» форми менеджера

На цьому вікні менеджер може виконувати такі дії:

– переглядати інформацію у таблиці про кімнати для тварин, а саме їхні номери та тип тварин для яких призначена ця кімната;

– додавати, виключати та редагувати інформацію про доступні види тварин у області вікна «допустимі види тварин», вказуючи максимальну та мінімальну тривалість резервації для даного типу тварин та вказуючи ціну для кожного з трьох розмірів тварини;

– додавати та видаляти кімнати для тварин у області вікна «кімнати для тварин»;

– додавати та змінювати тип та посилання на трансляцію для обраної кімнати;

– переглядати трансляцію з камери обраної кімнати.

На рисунку 4.18 продемонстровано вікно «Контракти та права доступу» форми менеджера.

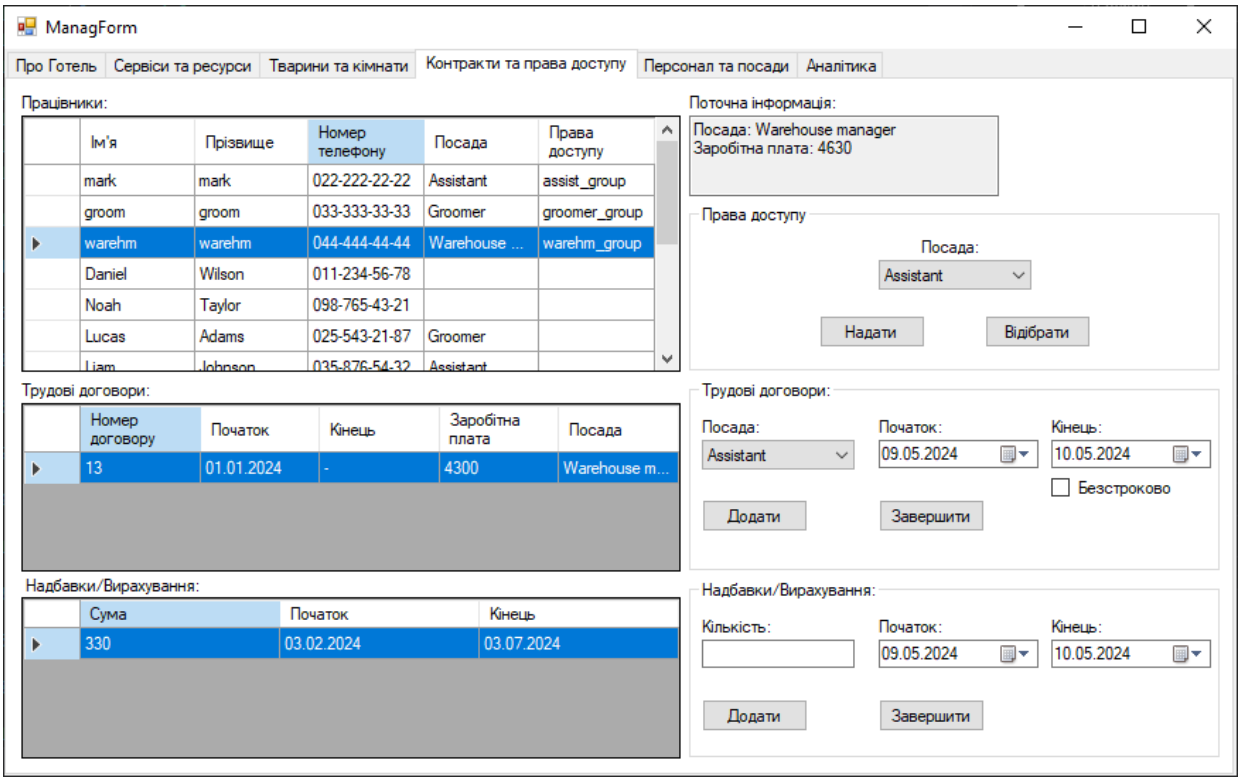


Рисунок 4.18 – Вікно «Контракти та права доступу» форми менеджера

На цьому вікні менеджер може виконувати такі дії:

– переглядати інформацію у таблиці «працівники» про зареєстрованих працівників, а саме їхню поточну посаду та права доступу;

– переглядати інформацію у таблиці «трудові договори» про трудові договори обраного працівника;

– переглядати інформацію у таблиці «надбавки/вирахування» про надбавки/вирахування для обраного договору, обраного працівника;

– надавати на віднімати права доступу на обрану посаду від обраного в таблиці працівника у області вікна «права доступу»;

– додавати новий або завершувати обраний у таблиці договорів трудовий договір у області вікна «трудові договори»;

– додавати надбавку/вирахування до обраного трудового договору у області вікна «надбавки/вирахування».

На рисунку 4.19 продемонстровано вікно «Персонал та посади» форми менеджера.

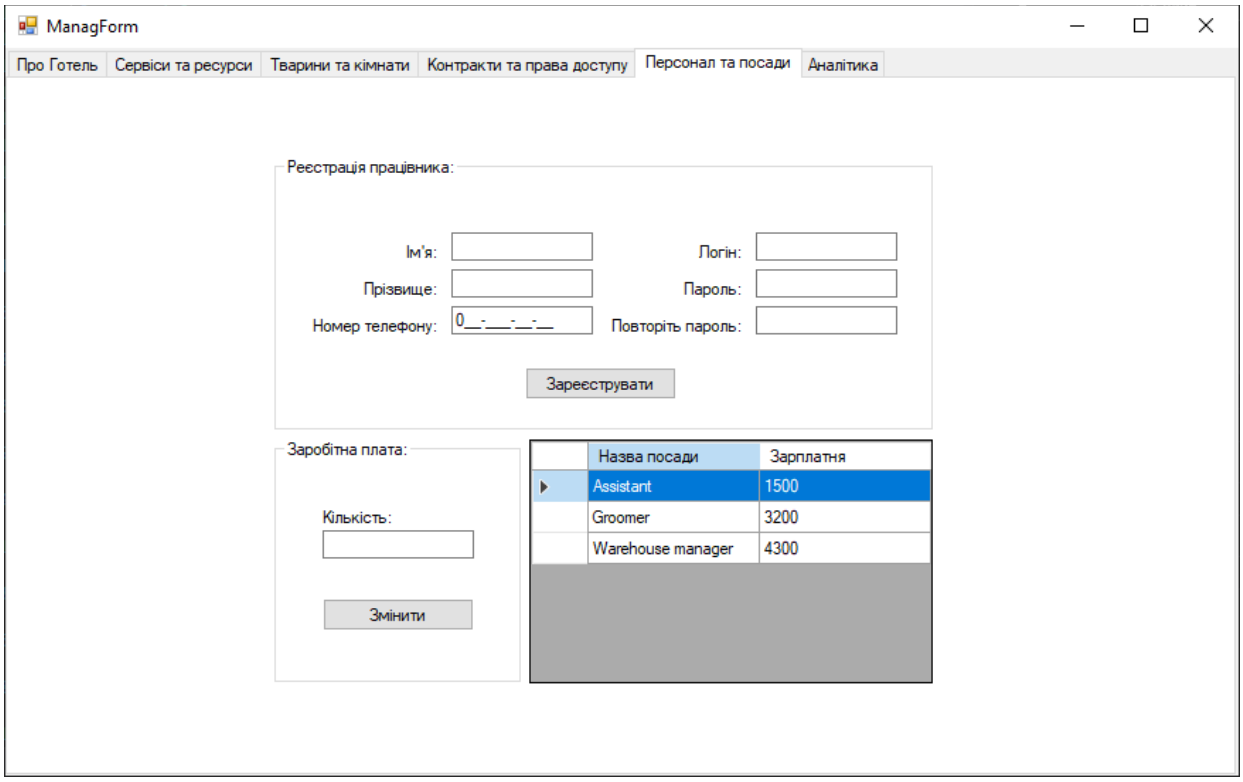


Рисунок 4.19 – Вікно «Персонал та посади» форми менеджера

На цьому вікні менеджер може виконувати такі дії:

– переглядати інформацію про зарплатню відповідної посади у таблиці;

– реєструвати нового працівника у області вікна «реєстрація працівника»;

– змінювати заробітну платню для обраної у таблиці посади у області вікна «заробітна плата».

Нижче буде наведено вікно форми менеджера «Аналітика» з представленням всіх наявних аналітичних функцій (див. рисунки 4.20 - 4.26).

На рисунку 4.20 продемонстрована аналітична функція «Рейтинг клієнтів». Ця аналітична функція відображає рейтинг клієнтів за кількості відвідувань готелю, кількості зареєстрованих вихованців, середньої кількості витрат на резервацію та сумарну кількість витрат.

На рисунку 4.21 продемонстрована аналітична функція «Дохід за рік». Ця аналітична функція відображає дохід готелю за обраний менеджером рік у вигляді графіку.

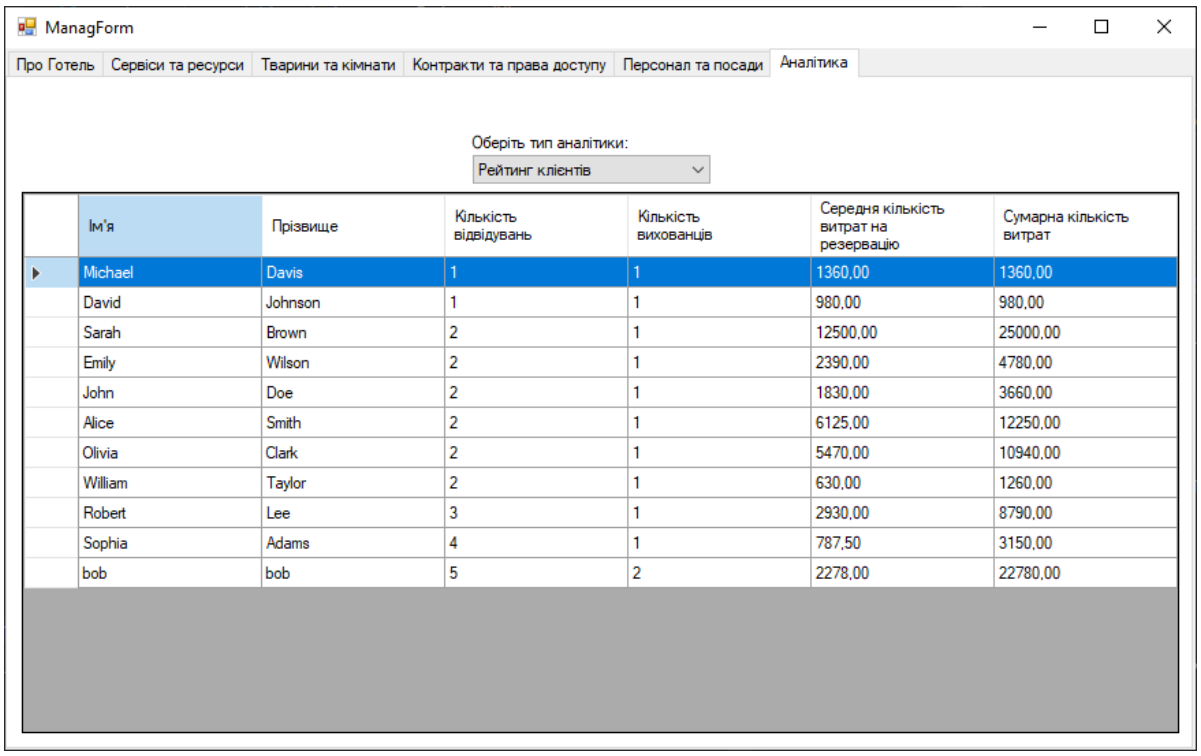
На рисунку 4.22 продемонстрована аналітична функція «Популярні сервіси». Ця аналітична функція відображає популярність кожного сервісу (кількість разів, коли цей сервіс був обраний) за обраний менеджером проміжок часу.

На рисунку 4.23 продемонстрована аналітична функція «Витрачені ресурси». Ця аналітична функція відображає сумарну кількість витрат кожного ресурсу за обраний менеджером проміжок часу.

На рисунку 4.24 продемонстрована аналітична функція «Доставлені ресурси». Ця аналітична функція відображає всі звіти про доставлені ресурси за обраний менеджером проміжок часу.

На рисунку 4.25 продемонстрована аналітична функція «Аналіз роботи асистентів». Ця аналітична функція відображає кількість прийнятих резервацій та їх сумарний прибуток для готелю для кожного асистента за обраний менеджером проміжок часу.

На рисунку 4.26 продемонстрована аналітична функція «Відгуки клієнтів». Ця аналітична функція відображає відгуки клієнтів, оцінку та інформацію про клієнтів.



Ім'я	Прізвище	Кількість відвідувань	Кількість вихованців	Середня кількість витрат на резервацію	Сумарна кількість витрат
Michael	Davis	1	1	1360,00	1360,00
David	Johnson	1	1	980,00	980,00
Sarah	Brown	2	1	12500,00	25000,00
Emily	Wilson	2	1	2390,00	4780,00
John	Doe	2	1	1830,00	3660,00
Alice	Smith	2	1	6125,00	12250,00
Olivia	Clark	2	1	5470,00	10940,00
William	Taylor	2	1	630,00	1260,00
Robert	Lee	3	1	2930,00	8790,00
Sophia	Adams	4	1	787,50	3150,00
bob	bob	5	2	2278,00	22780,00

Рисунок 4.20 – Аналітична функція «Рейтинг клієнтів» форми менеджера

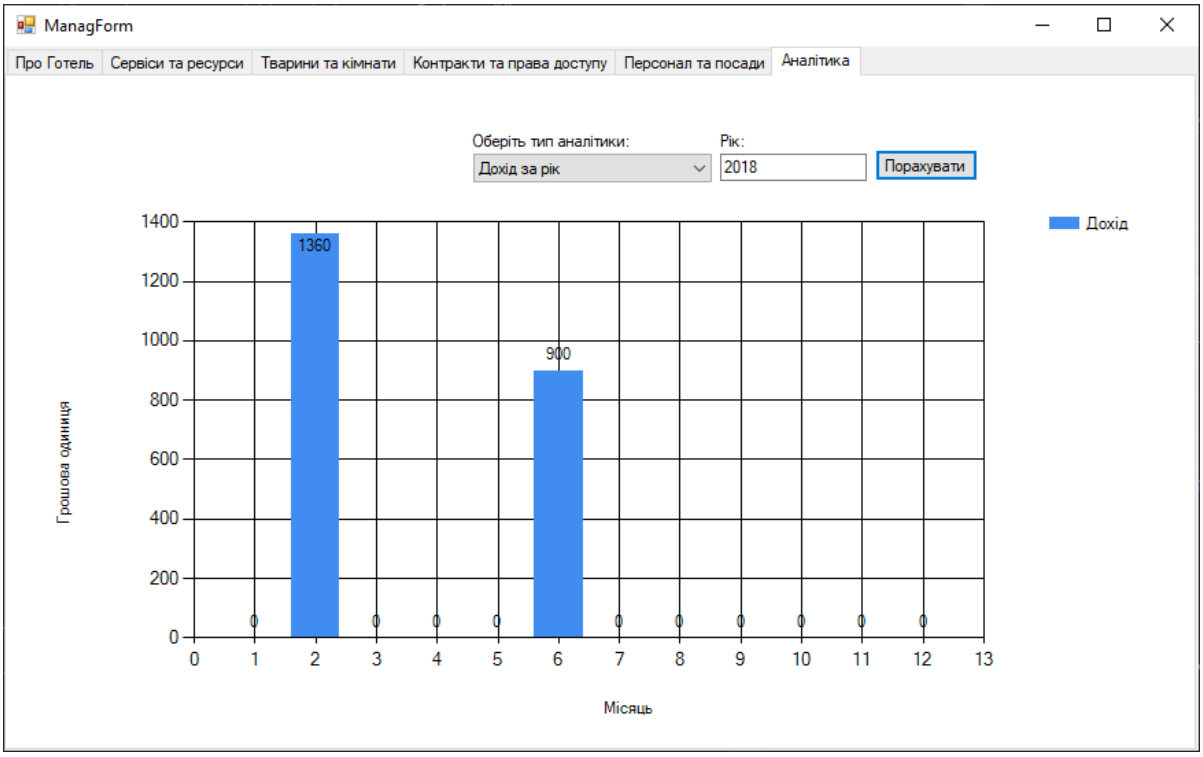


Рисунок 4.21 – Аналітична функція «Дохід за рік» форми менеджера

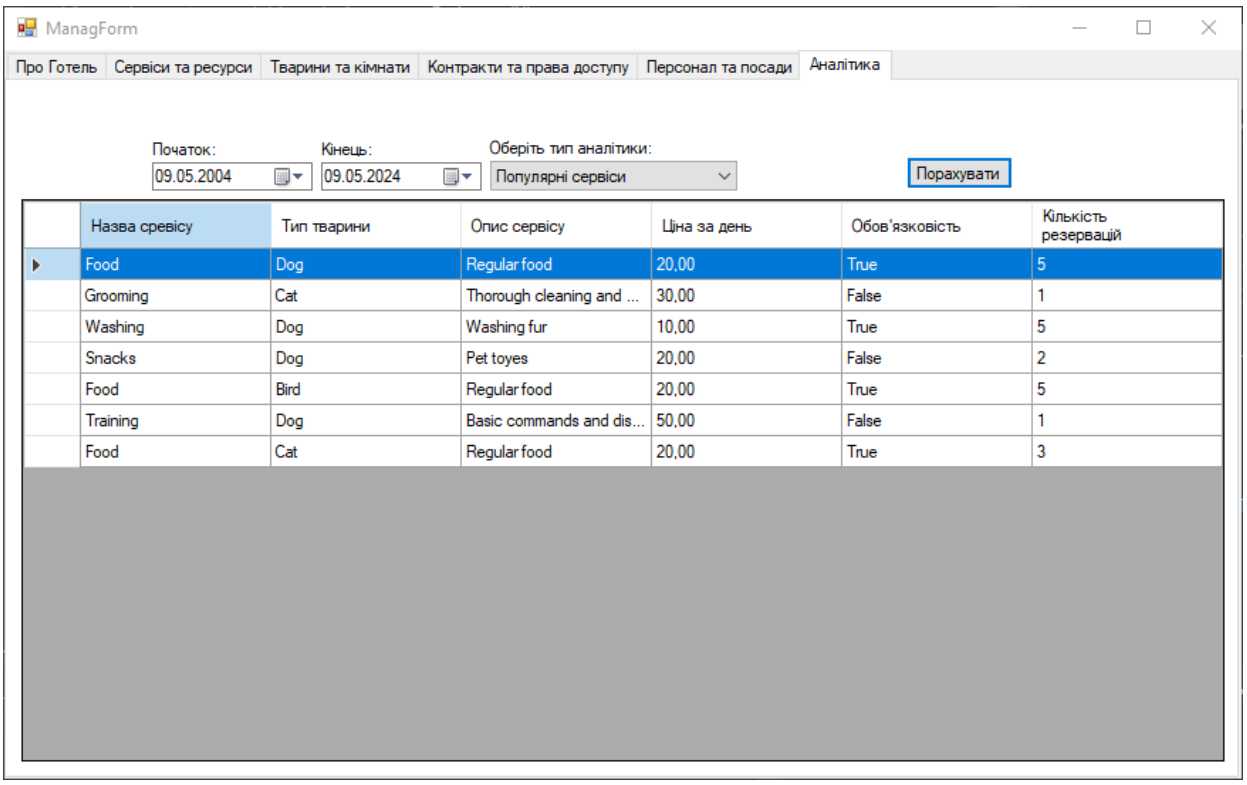


Рисунок 4.22 – Аналітична функція «Популярні сервіси» форми менеджера

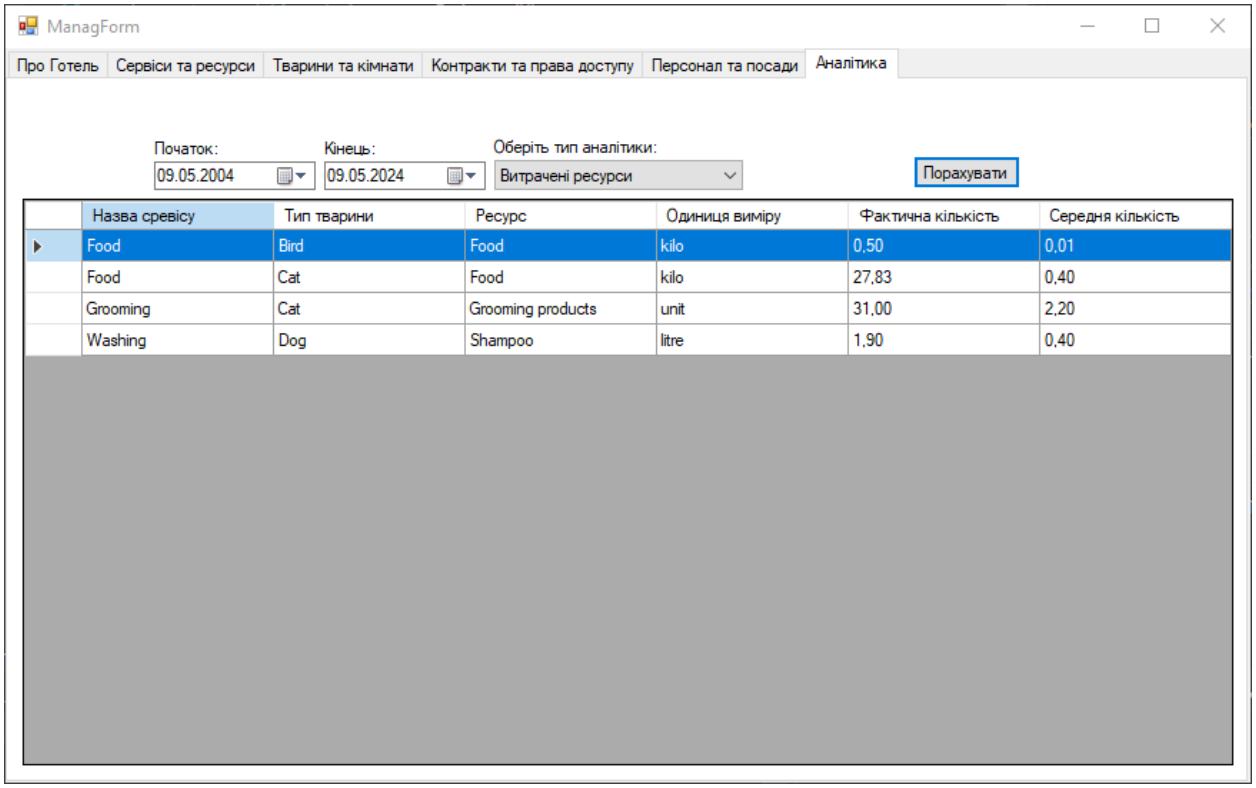


Рисунок 4.23 – Аналітична функція «Витрачені ресурси» форми менеджера

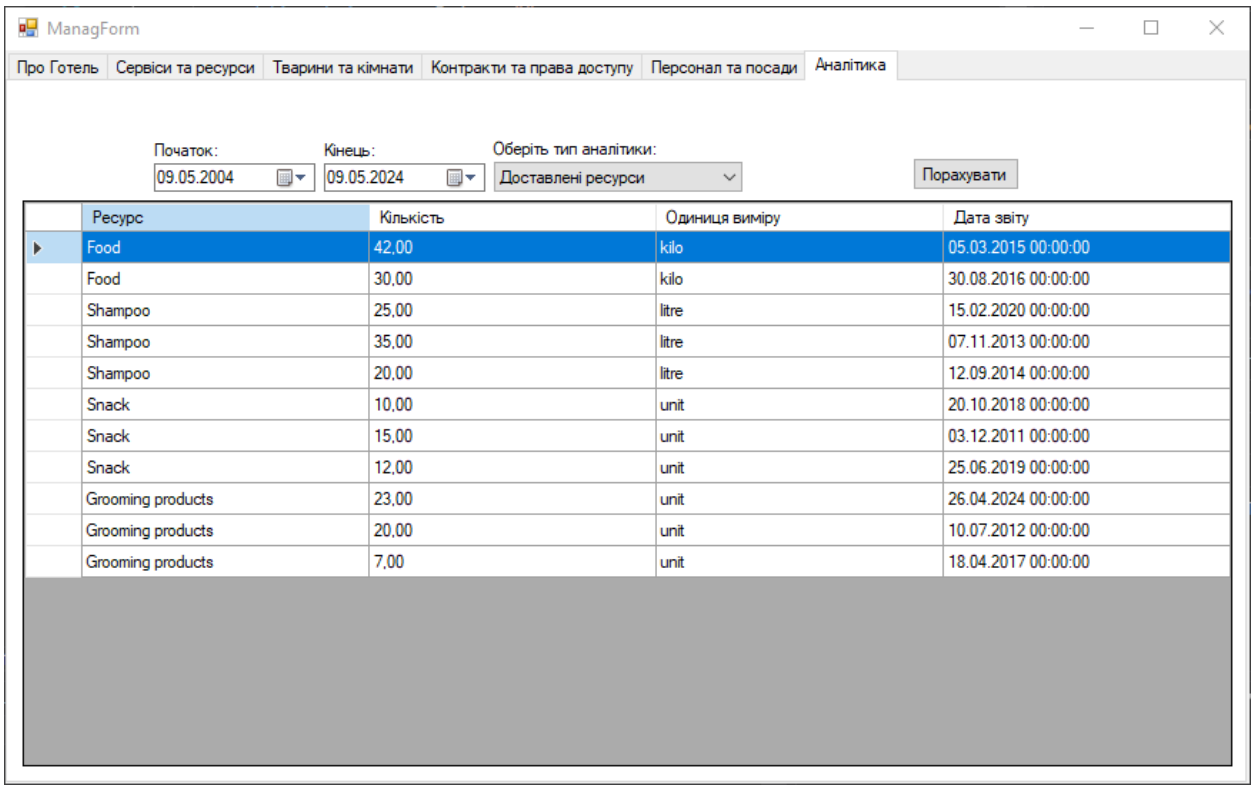


Рисунок 4.24 – Аналітична функція «Доставлені ресурси» форми менеджера

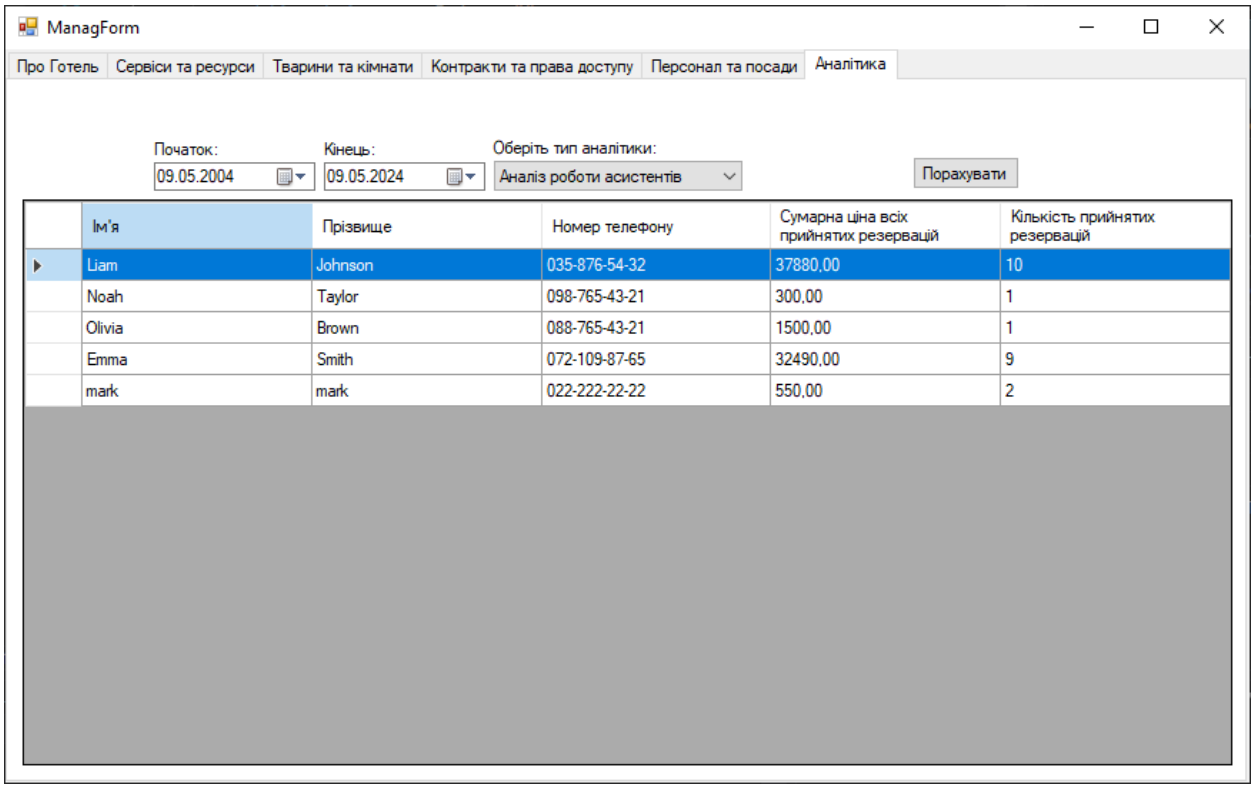


Рисунок 4.25 – Аналітична функція «Аналіз роботи асистентів» форми менеджера

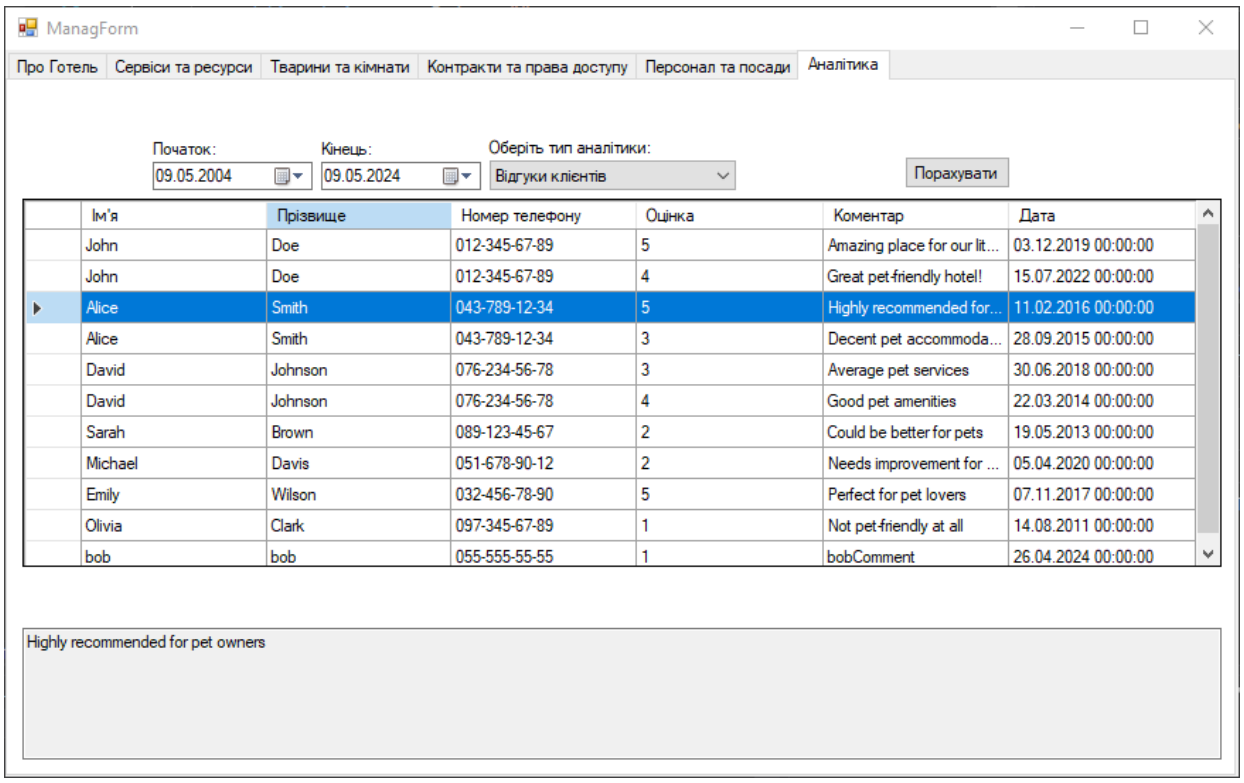


Рисунок 4.26 – Аналітична функція «Відгуки клієнтів» форми менеджера

4.7 Інтерфейс ветеринара

На рисунку 4.27 продемонстровано вікно «Мій профіль» форми ветеринара, яке є початковим вікном, на яке потрапляє ветеринар, після авторизації.

На цьому вікні ветеринар може виконувати такі дії:

– переглядати та змінювати свої персональні дані у області вікна «мої дані»;

– переглядати свої трудові договори у таблиці «трудові договори»;

– переглядати свої надбавки/вирахування у таблиці «надбавки/вирахування»;

– переглядати свою поточну робочу інформацію у області вікна «поточна інформація».

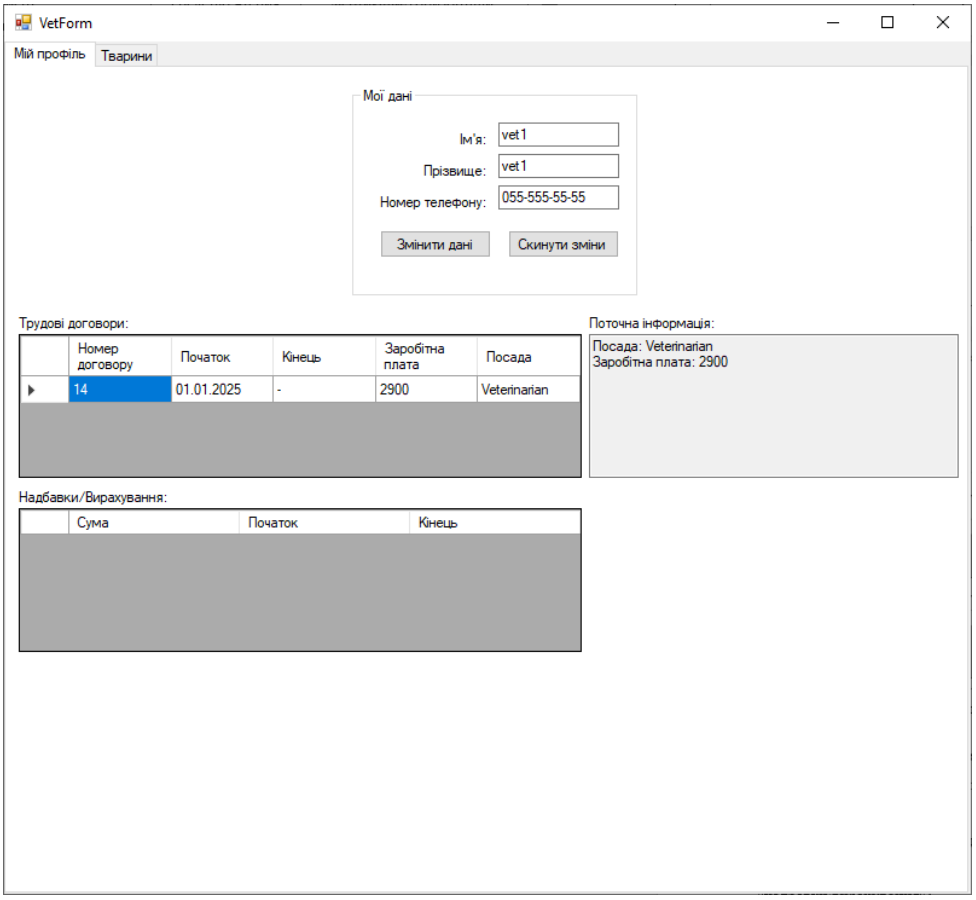


Рисунок 4.27 – Вікно «Мій профіль» форми ветеринара

На рис. 4.28 продемонстровано вікно «Тварини» форми ветеринара.

На цьому вікні ветеринар може виконувати такі дії:

– переглядати, змінювати, видаляти типи аналізів та нормативні значення цих аналізів для кожного типу тварин у області вікна «Типи аналізів»;

– переглядати та змінювати призначення ліків для обраного вихованця (за умовою, що цей вихованець призначений на поточного ветеринара) у області вікна «Призначення ліків»;

– переглядати опис проблем зі здоров’ям обраного вихованця у області вікна «Опис проблем зі здоров’ям»;

– переглядати інформацію про тварин, які перебувають у готелі та потребують медичного догляду у таблиці «Тварини, яким потрібен медичний догляд»;

– переглядати інформацію про історію аналізів обраного вихованця у таблиці «Хронологія аналізів»;

– переглядати інформацію про історію прийому медичних препаратів обраного вихованця у таблиці «Хронологія прийому медичних препаратів»;

– додавати інформацію про новий аналіз обраного вихованця у області вікна «Додати новий аналіз»;

– сортувати всіх наявних вихованців у таблиці «Тварини, яким потрібен медичний догляд» у області вікна «Додати новий аналіз» за такими фільтрами: «Всі», «Призначені на мене», «Очікують призначення»;

– переглядати відео трансляцію з кімнати обраного вихованця у області вікна «Відео з камери спостереження».

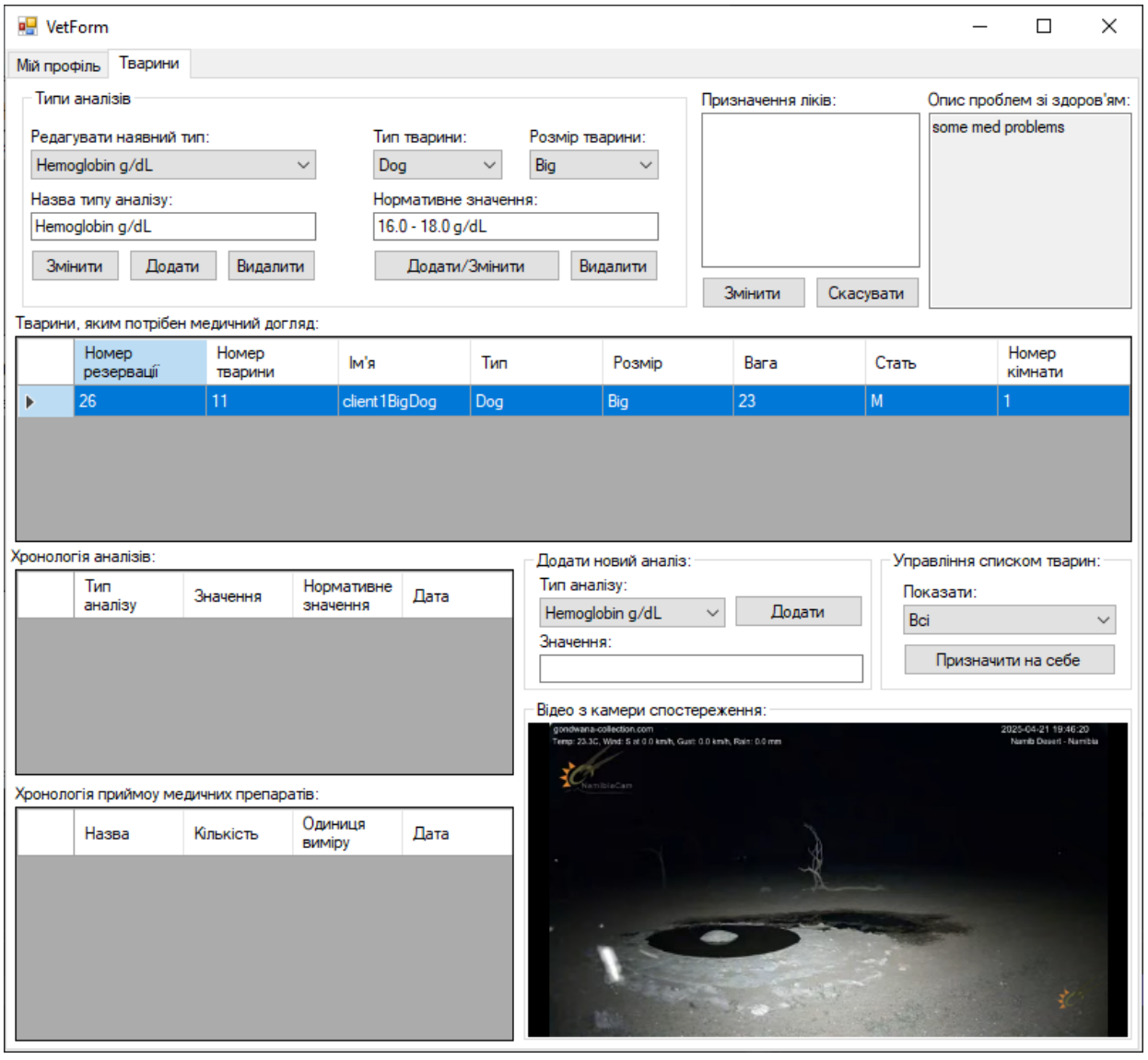


Рисунок 4.28 – Вікно «Тварини» форми ветеринара

У мобільному застосунку ветеринар може дивитися трансляції з камер вихованців, які знаходяться у готелі на поточний момент та потребують медичного догляду (рис. 4.29).

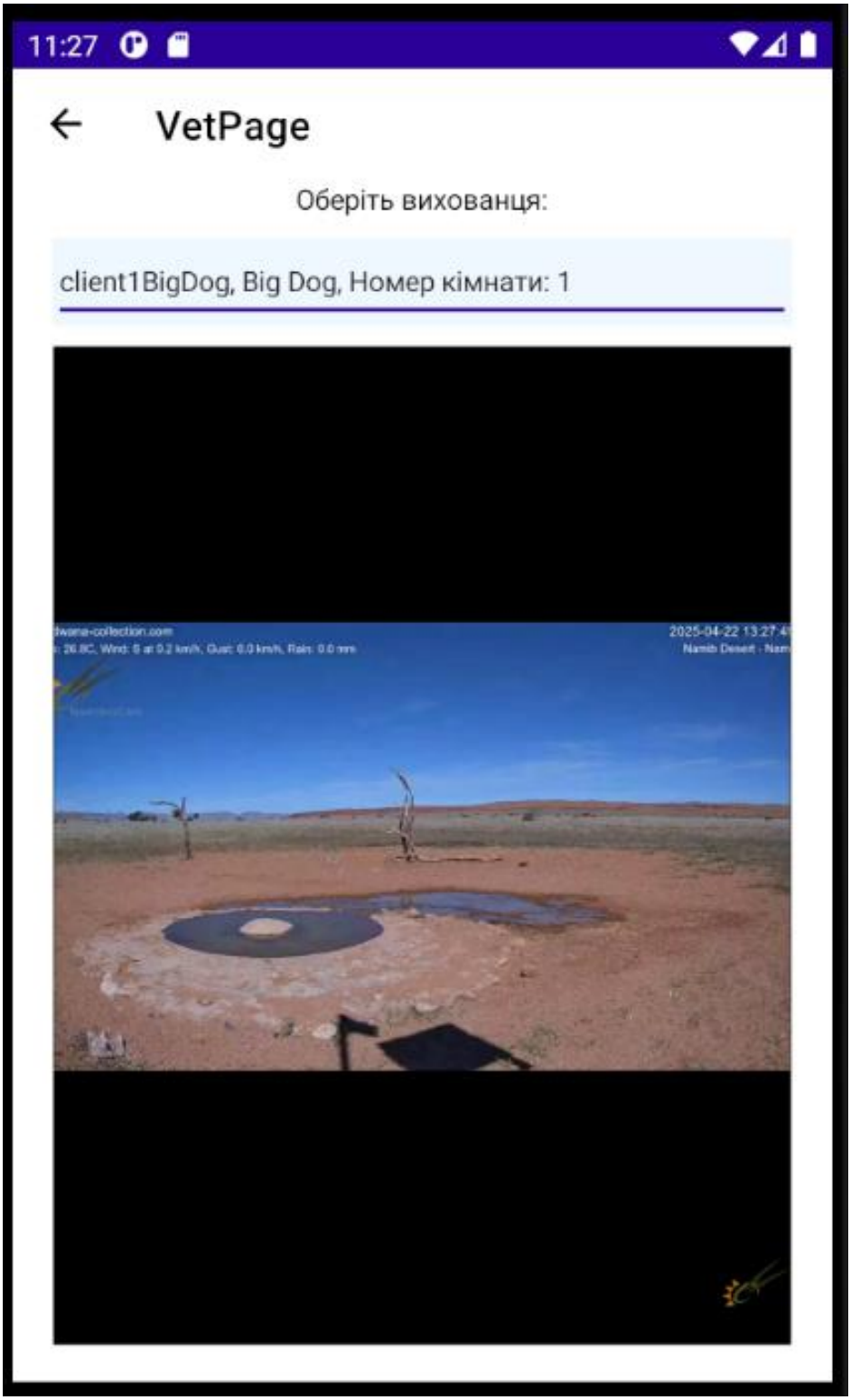


Рисунок 4.29 – Форма ветеринара у мобільному застосунку

ВИСНОВКИ

В результаті аналізу предметної області (ПрО) сформульована мета створення ІС проектування і реалізація ІС готелю для тварин, визначений список категорій її користувачів (Клієнт, Асистент, Грумер, Завідувач складу, Менеджер, Ветеринар) і задач, які користувачі повинні розв'язувати за допомогою ІС. Сформовані вимоги до даних і спроектована база даних з 21 таблиць для зберігання і маніпулювання даними ПрО.

Для створення ІС обрані дворівнева архітектура клієнт-сервер, шаблон проектування MVP, СУБД PostgreSQL, мова програмування C# і Windows Forms для розробки користувацького інтерфейсу дестктопного застосунку. Для розробки мобільного застосунку обрана платформа .NET MAUI. Також, для зв'язування користувацького застосунку з БД, були використані методи зовнішньої бібліотеки Npgsql. Реалізована система відеоспостереження за вихованцями у десктопному та мобільному застосунках.

Інтерфейс клієнтської програми розроблений з урахуванням вимог всіх користувачів і надає необхідний функціонал для розв'язання відповідних задач. Доступ до даних з боку різних категорій користувачів розмежований за допомогою механізму ролей і привілеїв. Захист від несанкціонованого доступу реалізований шляхом використання механізму аутентифікації і авторизації. За рахунок використання в створеній ІС дворівневої архітектури клієнт-сервер досягнута стабільність і надійність системи, а завдяки використанню шаблону проектування MVP – її висока масштабованість.

Створена ІС є добре структурованою, в ній можливо як нарощування функціоналу користувачів вже наявних категорій, наприклад, введення менеджером сезонних акцій на обрані сервіси або обрані типи тварин, так і додавання функціоналу нових категорій користувачів, наприклад, працівників відділу маркетингу, які аналізують популярні товари серед клієнтів, завдяки можливості інтеграції системи магазину готелю, та створюють нову або поліпшують існуючу продукцію.

Опробація проекту виконана на двадцять другій всеукраїнській конференції студентів і молодих науковців [1].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гавинський І.А., Малахов Є.В. Інформаційна система готелю для тварин // Інформатика, інформаційні системи та технології; тези доповідей двадцять другої всеукраїнської конференції студентів і молодих науковців – Одеса, 25 квітня 2025 р. – Одеса, 2025. – с. 191-193

2. Cloudbeds. Hotel Management Software & PMS System [Електронний ресурс] – Режим доступу: https://www.cloudbeds.com – Дата звернення: 21.04.2025.

3. Laika. Готельна система управління та бронювання [Електронний ресурс] – Режим доступу: https://laika.od.ua – Дата звернення: 21.04.2025.

4. Рачинська А.Л. Конспект лекцій з дисципліни «Об’єктно-орієнтоване програмування» для здобувачів вищої освіти першого (бакалаврського) рівня. Електронне видання. – Одеса: ОНУ, 2021.

5. PostgreSQL 16.3 Documentation [Електронний ресурс] // The PostgreSQL Global Development Group. – 1996-2024. – Режим доступу: https://www.postgresql.org/docs/16/index.html.

6. Рачинська А.Л. Конспект лекцій з дисципліни «Комп’ютерна графіка» для здобувачів вищої освіти першого (бакалаврського) рівня. Електронне видання. – Одеса: ОНУ, 2021.

7. .NET MAUI Documentation [Електронний ресурс] // Microsoft Learn. – 2025. – Режим доступу до ресурсу: https://learn.microsoft.com/ru-ru/dotnet/maui/?view=net-maui-9.0 – Дата звернення: 21.04.2025.

8. LibVLCSharp [Електронний ресурс] // VideoLAN. – 2025. – Режим доступу: https://code.videolan.org/videolan/LibVLCSharp – Дата звернення: 21.04.2025.

9. Малахов Є.В. Організація баз даних: конспект лекцій // Є.В. Малахов / Одеса: Одес. нац. ун-т ім. І. І. Мечникова, 2020. – 206 с.

10. Рачинська А.Л., Царенко О.П. Конспект лекцій з дисципліни «Структури даних та Алгоритми» для здобувачів вищої освіти першого (бакалаврського) рівня. Електронне видання. — Одеса: ОНУ, 2021.

ДОДАТОК А

Сценарії користувачів ІС «Готель для тварин»

Менеджер:

– я хочу мати можливість змінювати зарплатню, тривалість трудових договорів, надбавки чи штрафи працівників щоб контролювати фінансові витрати на персонал при контролі якості роботи персоналу;

– я хочу змінювати інформацію щодо сервісів, ресурсів, кімнат та доступних для резервації типів тварин щоб контролювати ціни та обмеження на тварин і сервіси, які готель надає клієнтам при коригуванні та аналізу витрат на обслуговування готелю;

– я хочу мати можливість переглядати звіти про використання ресурсів та популярності сервісів серед клієнтів щоб наочно подивитися на результати роботи готелю та підвести підсумки за деякий період часу при прогнозуванні, коригуванні цін сервісів і правил готелю.

Асистент:

– я хочу мати можливість оформляти, підтверджувати та видаляти заявки від клієнтів щоб швидко та зручно маніпулювати великою кількістю заявок при роботі з заявками від клієнтів;

– я хочу мати можливість реєструвати нових клієнтів і їх вихованців щоб дистанційно допомагати клієнтам реєструватися у застосунку та реєструвати їхніх тварин при роботі з клієнтами, які мають труднощі з самостійним процесом реєстрації;

– я хочу мати можливість переглядати інформацію про клієнта та його резервації щоб нагадувати клієнтам через дзвінки або SMS-повідомлення про закінчення резервації їхніх вихованців при закінченні терміну резервації тварини;

– я хочу мати можливість переглядати інформацію про готель щоб надавати клієнтам інформацію, яка їх цікавить під час прийому дзвінків від клієнтів.

Грумер:

– я хочу мати можливість переглядати інформацію про тварин у готелі для завчасної підготовки відповідних ресурсів та інструментів для догляду за тваринами в процесі догляду за тваринами.

– я хочу мати можливість переглядати інформацію про підключені сервіси для вихованця для підготовки відповідних ресурсів для цих сервісів при наданні певного сервісу конкретній тварині.

Завскладом:

– я хочу мати можливість оформлювати звіти про витрачені ресурси, прив’язуючи витрату ресурсу до конкретного сервісу конкретної резервації щоб мати можливість відслідковувати загальні витрати готелю та залишки ресурсів на складі при надходженні заявок від ветеринарів та грумерів на використання того чи іншого ресурсу.

– я хочу мати можливість оформлювати звіти про ресурси, які були доставлені на склад щоб вести облік товару при прийомі ресурсів.

Клієнт:

– я хочу мати можливість переглядати опис сервісів та допустимих тварин готелю щоб знати правила та обмеження готелю під час ознайомлення з готелем та аналізом тривалості резервації та додатковими сервісами.

– я хочу реєструвати нових вихованців та резервації для них щоб мати можливість переглядати історію резервацій та заплановані резервації для кожної тварини, а також мати можливість дивитися на своїх вихованців через камери відеоспостереження при аналізі своїх витрат на резервацію та задля нагадування про заплановані резервації.

– я хочу мати можливість переглядати відгуки щоб дивитися думки інших клієнтів щодо якості роботи готелю при прийнятті рішення, щодо обрання додаткових сервісів або деталей оформлення резервації.

– я хочу мати можливість залишати відгуки щоб висловлювати свою думку про готель при закінченні дії резервації мого вихованця.

Ветеринар:

– я хочу мати можливість редагувати типи та нормативні значення аналізів для всіх типів тварин щоб мати можливість наочно бачити різницю між фактичними показниками та нормативними під час перевірки аналізів вихованців.

– я хочу мати можливість переглядати інформацію щодо вихованців та їх опис проблем зі здоров’ям щоб виписувати ефективні ліки та методи одужання під час розробки програми одужання для тварини.

– я хочу мати можливість бачити історію аналізів та прийому ліків, і трансляцію з кімнати тварини щоб слідкувати за станом вихованця, та, при необхідності, редагувати програму одужання під час відстеження здоров’я вихованця.

ДОДАТОК Б

Задачі користувачів ІС «Готель для тварин»

Таблиця Б.1 – Список задач користувачів ІС «Готель для тварин»

Номер	Задача	Вхідні дані	Вихідні дані
Менеджер			
М1	Переглянути інформацію про готель	Відсутні	Інформація про готель (сервіси, ресурси, кімнати, тварини, працівники, посади)
М2	Додати сервіс	Назва, опис сервісу, ціна	Новий сервіс
М3	Відключити сервіс	Назва сервісу	Відсутні
М4	Редагувати інформацію про сервіс	Назва сервісу	Оновлена інформація про сервіс
М5	Відключити сервіс для обраного типу тварин	Назва сервісу, тип тварини	Відсутні
М6	Редагувати інформацію про сервіс для обраного типу тварин	Назва сервісу, тип тварини	Оновлена інформація про сервіс для обраного типу тварин
М7	Додати витрату ресурсу на сервіс для обраного типу тварин	Назва сервісу, ресурсу та тварини, кількість ресурсу	Нова витрата ресурсу на сервіс для обраного типу тварин
М8	Редагувати витрату ресурсу на сервіс для обраного типу тварин	Назва сервісу, ресурсу та тварини, кількість ресурсу	Оновлена інформація про витрату ресурсу на сервіс для обраного типу тварин
М9	Видалити витрату ресурсу на сервіс для обраного типу тварин	Назва сервісу, ресурсу та тварини	Відсутні

Продовження таблиці Б.1

Номер	Задача	Вхідні дані	Вихідні дані
М10	Додати новий ресурс	Назва ресурсу, одиниця виміру, позначка медичного препарату	Новий ресурс
М11	Редагувати інформацію про ресурс	Назва ресурсу, одиниця виміру, позначка медичного препарату	Оновлена інформація про ресурс
М12	Видалити ресурс	Назва ресурсу	Відсутні
М13	Додати новий допустимий тип тварин	Назва, мінімальна та максимальна тривалість резервації, ціна за день для трьох розмірів тварини	Новий допустимий вид тварин
М14	Відключити допустимий тип тварин	Назва типу тварини	Відсутні
М15	Редагувати інформацію про допустимий тип тварин	Назва, мінімальна та максимальна тривалість резервації, ціна за день для трьох розмірів тварини	Оновлена інформація про допустимий тип тварин
М16	Додати нову кімнату для тварин	Тип кімнати, розмір тварин, які можуть перебувати у кімнаті, тип та посилання на трансляцію камери	Нова кімната
М17	Видалити кімнату для тварин	Номер кімнати	Відсутні
М18	Реєстрація у системі нового працівника	Ім’я, прізвище, номер телефону, логін, пароль	Новий працівник

Продовження таблиці Б.1

Номер	Задача	Вхідні дані	Вихідні дані
М19	Переглянути контракти, надбавки та вирахування обраного працівника	Табельний номер працівника	Інформація про контракти, надбавки та вирахування обраного працівника
М20	Додати новий трудовий договір	Табельний номер працівника, посада, початок та кінець дії договору	Новий трудовий договір
М21	Завершити трудовий договір	Номер договору	Відсутні
М22	Додати надбавку чи вирахування	Табельний номер працівника, кількість надбавки чи вирахування, початок та кінець дії надбавки чи вирахування	Нова надбавка чи вирахування
М23	Завершити дію надбавки чи вирахування	Номер надбавки чи вирахування	Відсутні
М24	Переглянути всі наявні посади	Відсутні	Назва та заробітна плата усіх посад працівників
М25	Змінити зарплатню для обраної посади	Назва посади, кількість зарплатні	Оновлена інформація про посаду
М26	Перегляд рейтингу клієнтів	Відсутні	Інформація про рейтинг клієнтів
М27	Перегляд доходу за рік	Рік	Інформація про дохід за обраний рік
М28	Перегляд популярних сервісів	Діапазон дат для перегляду	Інформація про популярні сервіси

Продовження таблиці Б.1

Номер	Задача	Вхідні дані	Вихідні дані
М29	Перегляд витрачених ресурсів	Діапазон дат для перегляду	Інформація про витрачені на сервіси ресурси
М30	Перегляд доставлених ресурсів	Діапазон дат для перегляду	Інформація про звіти доставлених ресурсів
М31	Перегляд аналізу роботи асистентів	Діапазон дат для перегляду	Інформація про ефективність роботи асистентів
М32	Перегляд відгуків клієнтів	Діапазон дат для перегляду	Інформація про відгуки клієнтів
Асистент			
А1	Переглянути інформацію про свої персональні дані	Відсутні	Персональні дані працівника
А2	Переглянути інформацію про свої трудові договори	Відсутні	Інформація про трудові договори
А3	Переглянути свої надбавки та вирахування	Відсутні	Інформація про надбавки та вирахування
А4	Змінити свої персональні дані	Ім’я, прізвище, номер телефону	Оновлена інформація про персональні дані
А5	Переглянути інформацію про готель	Відсутні	Інформація про готель (доступні сервіси, тварини)
А6	Переглянути інформацію про резервації	Відсутні	Інформація про підтвердженні та не підтвердженні резервації
А7	Переглянути інформацію про вихованця обраної резервації	Номер резервації	Інформація про ім’я, прізвище, номер телефону клієнта, ім’я та медичний опис тварини обраної резервації

Продовження таблиці Б.1

Номер	Задача	Вхідні дані	Вихідні дані
А8	Підтвердити заявку	Номер заявки	Оновлена інформація про заявку (підтверджена)
А9	Видалити заявку	Номер заявки	Відсутні
А10	Переглянути інформацію про обраного клієнта	Логін клієнта	Інформація про ім’я, прізвище, номер телефону обраного клієнта
А11	Зареєструвати нового клієнта	Ім’я, прізвище, номер телефону, логін, пароль	Новий зареєстрований клієнт
А12	Зареєструвати тварину для клієнта	Ім’я, тип, розмір, вага, стать тварини та логін клієнта	Новий зареєстрований вихованець
А13	Зареєструвати резервацію для клієнта	Номер вихованця, початок та кінець резервації, номери обраних додаткових сервісів, опціонально: медичний опис тварини	Нова резервація
Грумер			
Г1	Переглянути інформацію про свої персональні дані	Відсутні	Персональні дані працівника
Г2	Переглянути інформацію про свої трудові договори	Відсутні	Інформація про трудові договори
Г3	Переглянути свої надбавки та вирахування	Відсутні	Інформація про надбавки та вирахування
Г4	Змінити свої персональні дані	Ім’я, прізвище, номер телефону	Оновлена інформація про персональні дані

Продовження таблиці Б.1

Номер	Задача	Вхідні дані	Вихідні дані
Г5	Переглянути інформацію про резервації, що діють сьогодні	Відсутні	Інформація про резервації та тварин, що діють сьогодні та потребують обслуговування
Г6	Переглянути інформацію про підключені до обраної резервації сервіси	Номер резервації	Інформація про підключені до обраної резервації сервіси та ресурси, які потрібно витратити на ці сервіси
Завскладом			
Зв1	Переглянути інформацію про свої персональні дані	Відсутні	Персональні дані працівника
Зв2	Переглянути інформацію про свої трудові договори	Відсутні	Інформація про трудові договори
Зв3	Переглянути свої надбавки та вирахування	Відсутні	Інформація про надбавки та вирахування
Зв4	Змінити свої персональні дані	Ім’я, прізвище, номер телефону	Оновлена інформація про персональні дані
Зв5	Переглянути інформацію про наявні ресурси на складі	Відсутні	Інформація про наявні ресурси на складі та їх кількість
Зв6	Додати звіт про використаний ресурс на обраний сервіс обраної резервації	Номер резервації, назва сервісу, назва ресурсу, кількість ресурсу	Новий звіт про використаний ресурс на обраний сервіс обраної резервації
Зв7	Додати звіт про доставлений на склад ресурс	Назва ресурсу, кількість ресурсу	Новий звіт про доставлений на склад ресурс

Продовження таблиці Б.1

Номер	Задача	Вхідні дані	Вихідні дані
Клієнт			
К1	Переглянути інформацію про свої персональні дані	Відсутні	Персональні дані клієнта
К2	Змінити свої персональні дані	Ім’я, прізвище, номер телефону	Оновлена інформація про персональні дані
К3	Зареєструвати нову тварину	Ім’я, тип, розмір, стать, вага тварини	Новий зареєстрований вихованець
К4	Переглянути своїх зареєстрованих вихованців	Відсутні	Інформація про ім’я, тип, розмір, вагу вихованців та посилання на відео с камери кімнати
К5	Переглянути інформацію про готель	Відсутні	Інформація про готель (доступні сервіси, тварини)
К6	Перегляд відгуків клієнтів	Відсутні	Інформація про відгуки клієнтів (дата, оцінка та коментар)
К7	Переглянути інформацію про свої резервації	Відсутні	Інформація про резервації клієнта (номер резервації, ім’я тварини, кінець та початок резервації, підключені сервіси, ціна та статус заявки)

Продовження таблиці Б.1

Номер	Задача	Вхідні дані	Вихідні дані
К8	Відправити заявку на резервацію тварини	Номер вихованця, початок та кінець резервації, номери обраних додаткових сервісів, опціонально: медичний опис тварини	Нова заявка на резервацію обраної тварини
К9	Додати свій відгук	Оцінка, коментар, дата відгуку	Новий відкук
Ветеринар			
В1	Переглянути інформацію про свої персональні дані	Відсутні	Персональні дані працівника
В2	Переглянути інформацію про свої трудові договори	Відсутні	Інформація про трудові договори
В3	Переглянути свої надбавки та вирахування	Відсутні	Інформація про надбавки та вирахування
В4	Змінити свої персональні дані	Ім’я, прізвище, номер телефону	Оновлена інформація про персональні дані
В5	Додати новий тип аналізу	Назва типу аналізу	Доданий новий тип аналізу
В6	Змінити назву існуючого типу аналізу	Назва існуючого типу аналізу, нова назва	Змінена назва типу аналізу
В7	Видалити існуючий тип аналізу	Назва типу аналізу	Відсутні
В8	Додати або змінити нормативне значення для обраного типу аналізу для обраного типу тварини	Назва типу аналізу, тип та розмір тварини, нормативне значення	Інформація про нормативне значення обраного типу аналізу для обраного типу тварини

Продовження таблиці Б.1

Номер	Задача	Вхідні дані	Вихідні дані
B9	Переглянути інформацію про всіх тварин, які потребують медичного догляду	Відсутні	Інформація про тварин, які перебувають у готелі та потребують медичного догляду на поточний момент (номер резервації, номер, ім’я, тип, розмір, вага, стать, медичний опис проблеми зі здоров’ям тварини, номер кімнати, посилання на відео з камери у кімнаті)
B10	Переглянути інформацію про медичні препарати, які приймав вихованець у готелі	Номер тварини	Інформація про прийом медичних препаратів (назва, кількість та одиниця виміру препарату, дата прийому)
B11	Переглянути інформацію про всі результати аналізів обраного вихованця	Номер вихованця	Інформація про результати аналізів (назва типу аналізу, значення результату, нормативне значення, дата аналізу)

Продовження таблиці Б.1

Номер	Задача	Вхідні дані	Вихідні дані
B12	Призначити вихованця на себе	Номер резервації	Оновлена інформація про резервацію обраного вихованця (назначений ветеринар)
B13	Призначити ліки для вихованця	Номер резервації	Оновлена інформація про резервацію обраного вихованця (назначені ліки)
B14	Додати результат аналізу	Номер вихованця, тип та значення аналізу	Інформація про новий результат аналізу

ДОДАТОК В

Опис сутностей предметної області

Таблиця В.1 – Опис сутностей предметної області «Готель для тварин»

Ім’я атрибута	Призначення атрибута	Обмеження
Client (Клієнт)		
ClientNo	Номер клієнта	первинний ключ
Name	Ім’я клієнта	не порожнє
Surname	Прізвище клієнта	не порожнє
TelNo	Номер телефону клієнта	не порожнє, унікальне, значення повинно відповідати шаблону '0[0-9]{2}-[0-9]{3}-[0-9]{2}-[0-9]{2}'
Login	Логін клієнта	не порожнє, унікальне
Review (Відгук клієнта)		
ReviewNo	Номер відгуку клієнта	первинний ключ
ClientNo	Номер клієнта	не порожнє, зовнішній ключ для зв'язку з сутністю Client (ClientNo)
Rating	Оцінка клієнта	не порожнє, значення >= 1, значення <= 5
Comment	Зміст відгуку	
Date	Дата відгуку	не порожнє, значення <= сьогоднішній даті, за замовченням сьогоднішня дата
PetKind (Вид тварин)		
PetKindNo	Номер виду тварини	первинний ключ
KindTitle	Назва виду тварини	не порожнє, унікальне
MaxRentDuration	Максимальна кількість днів резервації для цього виду тварини	не порожнє, значення > 0, значення >= MinRentDuration
MinRentDuration	Мінімальна кількість днів резервації для цього виду тварини	не порожнє, значення > 0, значення <= MaxRentDuration

Продовження таблиці В.1

Ім’я атрибута	Призначення атрибута	Обмеження
ifActive	Стан цього виду тварини	не порожнє, значення з множини ('істина' (дозволений тип тварин), 'хиба' (заборонений тип тварин)), за замовчуванням 'істина'
AllowedPetType (Тип тварин за розміром)		
AllowedPetTypeNo	Номер типу тварини	первинний ключ
PetKindNo	Номер виду тварини	не порожнє, зовнішній ключ для зв'язку з сутністю PetKind (PetKindNo)
Size	Розмір тварини	не порожнє, значення з множини ('Small', 'Medium', 'Big')
PricePerDay	Ціна бронювання тварини цього типу за один день	не порожнє, значення > 0
		унікальна комбінація (PetKindNo, Size)
PetRoom (Кімната для тварини)		
PetRoomNo	Номер кімнати	первинний ключ
AllowedPetTypeNo	Номер типу тварини	не порожнє, зовнішній ключ для зв'язку з сутністю AllowedPetType (AllowedPetTypeNo)
StreamURL	Посилання на трансляцію камери у цій кімнаті	унікальне, за замовчування порожнє
StreamType	Тип трансляції	значення з множини ('YouTube', 'RTSP') якщо StreamURL не порожнє або значення порожнє якщо і StreamURL порожнє, за замовчування порожнє

Продовження таблиці В.1

Ім’я атрибута	Призначення атрибута	Обмеження
Pet (Вихованець клієнта)		
PetNo	Номер тварини	первинний ключ
ClientNo	Номер клієнта	не порожнє, зовнішній ключ для зв'язку з сутністю Client (ClientNo)
Name	Ім’я тварини	не порожнє
AllowedPetType	Номер типу тварини	не порожнє, зовнішній ключ для зв'язку з сутністю AllowedPetType (AllowedPetTypeNo)
Weight	Вага тварини	не порожнє, значення > 0
Gender	Стать тварини	не порожнє, значення з множини ('M', 'F')
Staff (Працівник готелю)		
StaffNo	Табельний номер працівника	первинний ключ
Name	Ім’я працівника	не порожнє
Surname	Прізвище працівника	не порожнє
TelNo	Номер телефону працівника	не порожнє, унікальне значення повинно відповідати шаблону '0[0-9]{2}-[0-9]{3}-[0-9]{2}-[0-9]{2}'
Login	Логін працівника	не порожнє, унікальне
Position (Посада)		
PositionNo	Номер посади	первинний ключ
PositionTitle	Назва посади	не порожнє, унікальне
Salary	Заробітна плата посади	не порожнє, значення > 0
StaffContract (Трудовий договір)		
StaffContractNo	Номер договору	первинний ключ
StaffNo	Табельний номер працівника	не порожнє, зовнішній ключ для зв'язку з сутністю Staff (StaffNo)

Продовження таблиці В.1

Ім’я атрибута	Призначення атрибута	Обмеження
PositionNo	Номер посади	не порожнє, зовнішній ключ для зв'язку з сутністю Position (PositionNo)
ContractStart	Дата початку дії договору	не порожнє
ContractFinish	Дата кінця дії договору	значення >= ContractStart (порожнє значення позначає, що це є безстроковим договором)
SalaryChange (Надбавка чи вирахування)		
SalaryChangeNo	Номер надбавки/вирахування	первинний ключ
StaffContractNo	Номер договору	не порожнє, зовнішній ключ для зв'язку з сутністю StaffContract (StaffContractNo)
Amount	Кількість надбавки/вирахування	не порожнє
DateStart	Дата початку дії надбавки/вирахування	не порожнє
DateFinish	Дата кінця дії надбавки/вирахування	не порожнє, значення >= DateStart
Reservation (Резервація)		
ReservationNo	Номер резервації	первинний ключ
PetNo	Номер тварини	не порожнє, зовнішній ключ для зв'язку з сутністю Pet (PetNo)
RentStart	Дата початку резервації	не порожнє
RentFinish	Дата кінця резервації	не порожнє, значення >= RentFinish
StaffNo	Табельний номер асистента	зовнішній ключ для зв'язку з сутністю Staff (StaffNo), за замовчуванням 'порожнє'

Продовження таблиці В.1

Ім’я атрибута	Призначення атрибута	Обмеження
PetRoomNo	Номер кімнати	зовнішній ключ для зв'язку з сутністю PetRoom (PetRoomNo), за замовчуванням 'порожнє'
MedDescription	Опис проблеми зі здоров’ям вихованця	за замовчуванням 'порожнє'
MedPrescription	Опис курсу лікування тварини	за замовчуванням 'порожнє'
VetStaffNo	Табельний номер ветеринара	зовнішній ключ для зв'язку з сутністю Staff (StaffNo), за замовчуванням 'порожнє'
Service (Сервіс)		
ServiceNo	Номер сервісу	первинний ключ
ServiceName	Назва сервісу	не порожнє, унікальне
Description	Опис сервісу	не порожнє
PricePerDay	Ціна сервісу за день	не порожнє, значення > 0
ServicePet (Відповідність сервісу до типу тварини)		
ServicePetNo	Номер відповідності	первинний ключ
ServiceNo	Назва сервісу	не порожнє, зовнішній ключ для зв'язку з сутністю Service (ServiceNo)
PetKindNo	Назва виду тварини	не порожнє, зовнішній ключ для зв'язку з сутністю PetKind (PetKindNo)
Required	Статус обов’язковості сервісу для виду тварини	не порожнє, значення з множини ('істина' (обов’язковий), 'хиба' (необов’язковий))
ifActive	Статус доступності сервісу для виду тварини	не порожнє, значення з множини ('істина' (доступний), 'хиба' (недоступний)), за замовчуванням 'істина'

Продовження таблиці В.1

Ім’я атрибута	Призначення атрибута	Обмеження
		унікальна комбінація (ServiceNo, PetKindNo)
Resourse (Ресурс)		
ResourseNo	Номер ресурсу	первинний ключ
Name	Назва ресурсу	не порожнє, унікальне
Unit	Одиниця виміру ресурсу	не порожнє
ifMedicine	Позначка медичного препарату	не порожнє, за замовчуванням 'хиба'
ServicePetResourse (Відповідність сервісу і тварини до ресурсу)		
ServicePetNo	Номер відповідності сервісу та тварини	не порожнє, зовнішній ключ для зв'язку з сутністю ServicePet (ServicePetNo)
ResourseNo	Номер ресурсу	не порожнє, зовнішній ключ для зв'язку з сутністю Resourse (ResourseNo)
AverageAmount	Середня кількість витраченого ресурсу за один день	не порожнє, значення > 0
		первинний ключ (ServiceNo, PetKindNo)
ReservService (Зарезервований сервіс)		
ReservServiceNo	Номер зарезервованого сервісу	первинний ключ
ReservationNo	Номер резервації	не порожнє, зовнішній ключ для зв'язку з сутністю Reservation (ReservationNo)
ServicePetNo	Номер відповідності сервісу та тварини	не порожнє, зовнішній ключ для зв'язку з сутністю ServicePet (ServicePetNo)
		унікальна комбінація (ReservationNo, ServicePetNo)
ProcurementReport (Звіт про доставлені ресурси)		
ProcurementReportNo	Номер звіту	первинний ключ

Продовження таблиці В.1

Ім’я атрибута	Призначення атрибута	Обмеження
ResourseNo	Номер ресурсу	не порожнє, зовнішній ключ для зв'язку з сутністю Resourse (ResourseNo)
ReportDate	Дата звіту	не порожнє, значення <= сьогоднішньої дати
Amount	Кількість ресурсу	не порожнє, значення > 0
ServiceCostReport (Звіт про витрачені ресурси)		
ServiceCostReportNo	Номер звіту	первинний ключ
ReservServiceNo	Номер зарезервованого сервісу	не порожнє, зовнішній ключ для зв'язку з сутністю ReservService (ReservServiceNo)
ResourseNo	Номер ресурсу	не порожнє, зовнішній ключ для зв'язку з сутністю Resourse (ResourseNo)
ReportDate	Дата звіту	не порожнє, значення <= сьогоднішньої дати
FactualAmount	Кількість ресурсу	не порожнє, значення > 0
AnalysisType (Тип аналізу)		
AnalysisTypeNo	Номер типу аналізу	первинний ключ
Name	Назва типу аналізу	не порожнє, унікальне
AnalysisPetType (Нормативне значення типу аналізу для типу тварини)		
AnalysisPetTypeNo	Номер відповідності типу аналізу і типу тварини	первинний ключ
AllowedPetTypeNo	Номер типу тварини	не порожнє, зовнішній ключ для зв'язку з сутністю AllowedPetType (AllowedPetTypeNo)
AnalysisTypeNo	Номер типу аналізу	не порожнє, зовнішній ключ для зв'язку з сутністю AnalysisType (AnalysisTypeNo)

Продовження таблиці В.1

Ім’я атрибута	Призначення атрибута	Обмеження
Normative	Нормативне значення аналізу	не порожнє
		унікальна комбінація (AllowedPetTypeNo, AnalysisTypeNo)
Analysis (Результат аналізу)		
AnalysisNo	Номер аналізу	первинний ключ
PetNo	Номер вихованця	не порожнє, зовнішній ключ для зв'язку з сутністю Pet (PetNo)
AnalysisPetTypeNo	Номер типу аналізу і типу тварини	не порожнє, зовнішній ключ для зв'язку з сутністю AnalysisPetType (AnalysisPetTypeNo)
Result	Значення результату аналізу	не порожнє
Date	Дата аналізу	не порожнє, значення <= сьогоднішньої дати, за замовчуванням ' сьогоднішня дата'

ДОДАТОК Г

Запити на створення таблиць бази даних

```
create table Client (
    ClientNo serial not null
        primary key,
    Name varchar not null,
    Surname varchar not null,
    TelNo varchar not null
        check (TelNo similar to '0[0-9]{2}-[0-9]{3}-[0-9]{2}-[0-
9]{2}')
        unique,
    Login varchar not null
        unique
    );
create table Review (
    ReviewNo serial not null
        primary key,
    ClientNo int not null
        references Client on update cascade,
    Rating smallint not null
        check (Rating between 1 and 5),
    Comment varchar,
    Date date not null
        check(Date <= current_date)
        default current_date
    );
create table PetKind (
    PetKindNo serial not null
        primary key,
    KindTitle varchar not null
        unique,
    MaxRentDuration int not null
        check (MaxRentDuration > 0 and MaxRentDuration >=
MinRentDuration),
    MinRentDuration int not null
        check (MinRentDuration > 0 and MinRentDuration <=
MaxRentDuration),
    ifActive bool not null default TRUE
    );
create table AllowedPetType (
    AllowedPetTypeNo serial not null
        primary key,
    PetKindNo int not null
        references PetKind on update cascade on delete cascade,
    Size varchar not null
        check (Size in ('Small', 'Medium', 'Big')),
    PricePerDay numeric(9,2) not null
        check(PricePerDay > 0),
```

Лістинг Г.1 – Запити на створення таблиць бази даних

```
    unique(PetKindNo, Size)
    );
create table PetRoom (
    PetRoomNo serial not null
        primary key,
    AllowedPetTypeNo int not null
        references AllowedPetType on update cascade,
    StreamURL varchar
        default null
        unique,
    StreamType varchar
        check (
                (StreamType in ('YouTube', 'RTSP') and StreamURL
is not null)
               or
                (StreamType is null and StreamURL is null)
            )
        default null
    );
create table Pet (
    PetNo serial not null
        primary key,
    ClientNo int not null
        references Client on update cascade,
    Name varchar not null,
    AllowedPetTypeNo int not null
        references AllowedPetType on update cascade on delete
cascade,
    Weight numeric(9,2) not null
        check (Weight > 0),
    Gender varchar not null
        check (Gender in ('M', 'F'))
    );
create table Staff (
    StaffNo serial not null
        primary key,
    Name varchar not null,
    Surname varchar not null,
    TelNo varchar not null
        check (TelNo similar to '0[0-9]{2}-[0-9]{3}-[0-9]{2}-[0-
9]{2}')
        unique,
    Login varchar not null
        unique
    );
create table Position (
    PositionNo serial not null
        primary key,
    PositionTitle varchar not null
        unique,
    Salary numeric(9,2) not null
```

Лістинг Г.1, аркуш 2

```
        check(Salary > 0)
    );
create table StaffContract (
    StaffContractNo serial not null
        primary key,
    StaffNo int not null
        references Staff on update cascade,
    PositionNo int not null
        references Position on update cascade,
    ContractStart date not null,
    ContractFinish date
        check (ContractFinish >= ContractStart)
    );
create table SalaryChange (
    SalaryChangeNo serial not null
        primary key,
    StaffContractNo int not null
        references StaffContract on update cascade on delete
cascade,
    Amount numeric(9,2) not null,
    DateStart date not null,
    DateFinish date not null
        check (DateFinish >= DateStart)
    );
create table Reservation (
    ReservationNo serial not null
        primary key,
    PetNo int not null
        references Pet on update cascade on delete cascade,
    RentStart date not null,
    RentFinish date not null check(RentFinish >= RentStart),
    StaffNo int
        references Staff on update cascade on delete set default
        default null,
    PetRoomNo int
        references PetRoom on update cascade on delete set
default
        default null,
    MedDescription varchar
        default null,
    MedPrescription varchar
        default null,
    VetStaffNo int
        references Staff on update cascade on delete set default
        default null
    );
create table Service (
    ServiceNo serial not null
        primary key,
    ServiceName varchar not null
        unique,
```

Лістинг Г.1, аркуш 3

```
    Description varchar not null,
    PricePerDay numeric(9,2) not null
        check(PricePerDay > 0)
    );
create table ServicePet (
    ServicePetNo serial not null
        primary key,
    ServiceNo int not null
        references Service on update cascade on delete cascade,
    PetKindNo int not null
        references PetKind on update cascade on delete cascade,
    Required bool not null,
    ifActive bool not null default TRUE,
    unique(ServiceNo, PetKindNo)
    );
create table Resourse (
    ResourseNo serial not null
        primary key,
    Name varchar not null
        unique,
    Unit varchar not null,
    ifMedicine bool not null
        default FALSE
    );
create table ServicePetResourse (
    ServicePetNo int not null
        references ServicePet on update cascade on delete
cascade,
    ResourseNo int not null
        references Resourse on update cascade on delete cascade,
    AverageAmount numeric(9,2) not null
        check (AverageAmount > 0),
    primary key (ServicePetNo, ResourseNo)
    );
create table ReservService (
    ReservServiceNo serial not null
        primary key,
    ReservationNo int not null
        references Reservation on update cascade on delete
cascade,
    ServicePetNo int not null
        references ServicePet on update cascade on delete
cascade,
    unique(ReservationNo, ServicePetNo)
    );
create table ProcurementReport (
    ProcurementReportNo serial not null
        primary key,
    ResourseNo int not null
        references Resourse on update cascade,
    ReportDate date not null
```

Лістинг Г.1, аркуш 4

```
        check(ReportDate <= current_date),
    Amount numeric(9,2) not null
        check(Amount > 0)
    );
create table ServiceCostReport (
    ServiceCostReportNo serial not null
        primary key,
    ReservServiceNo int not null
        references ReservService on update cascade on delete
cascade,
    ResourseNo int not null
        references Resourse on update cascade,
    ReportDate date not null
        check(ReportDate <= current_date),
    FactualAmount numeric(9,2) not null
        check(FactualAmount > 0));
create table AnalysisType (
    AnalysisTypeNo serial not null
        primary key,
    Name varchar not null
        unique);
create table AnalysisPetType (
    AnalysisPetTypeNo serial not null
        primary key,
    AllowedPetTypeNo int not null
        references AllowedPetType on update cascade,
    AnalysisTypeNo int not null
        references AnalysisType on update cascade,
    Normative varchar not null,
    unique(AllowedPetTypeNo, AnalysisTypeNo));
create table Analysis (
    AnalysisNo serial not null
        primary key,
    PetNo int not null
        references Pet on update cascade on delete cascade,
    AnalysisPetTypeNo int not null
        references AnalysisPetType on update cascade,
    Result varchar not null,
    Date date not null
        check(Date <= current_date)
        default current_date);
```

Лістинг Г.1, аркуш 5

ДОДАТОК Д

Запити на створення представлень бази даних

```
--Представлення, про всі допустимі типи тварин (ціна за день
СУМУМУЄТЬСЯ з ціною обов'язкових сервісів)
--drop view AllPetTypesInfo;
create or replace view AllPetTypesInfo as
--Цей підзапит виводить номер типу тварини та її вартість
ВКЛЮЧАючи обов'язкові сервіси
with PetTypeCost as
(
    select
    aPT.AllowedPetTypeNo,
    SUM(s.PricePerDay) as petTypeCost
    from
    PetKind pK inner join AllowedPetType aPT using (PetKindNo)
               inner join ServicePet sP using (PetKindNo)
               inner join Service s using (ServiceNo)
    where
    sP.Required = 'true' and sP.ifActive = true
    group by(aPT.AllowedPetTypeNo)
    order by aPT.AllowedPetTypeNo
)
select
KindTitle,
Size,
--Якщо у цього типу тварини немає обов'язкових сервісів, то
sP.petTypeCost + aPT.PricePerDay = null
--щоб цього уникнути, ми використовуємо COALESCE, яка повертає
перший не «null» аргумент
(COALESCE(sP.petTypeCost  + aPT.PricePerDay, aPT.PricePerDay))
as petTypeCost,
MaxRentDuration,
MinRentDuration
from
PetKind pK inner join AllowedPetType aPT using (PetKindNo)
           left outer join PetTypeCost sP using
(AllowedPetTypeNo)
where pK.ifActive = 'true'
;
--Представлення, про всі ServicePet
--drop view AllServicePets;
create or replace view AllServicePets as
select s.ServiceNo, KindTitle, ServiceName, Description,
PricePerDay, Required
from
Service s inner join ServicePet sP using(ServiceNo)
          inner join PetKind pK using(PetKindNo)
where sP.ifActive = 'true' and pK.ifActive = 'true';
```

Лістинг Д.1 – Запити на створення представлень бази даних

```
--Представлення, про вихованців клієнта
--drop view ClientPets;
create or replace view ClientPets as
with AllClientPets as
(
    select Pet.PetNo, Pet.Name, pK.KindTitle, aPT.Size,
Pet.Weight
    from
    Pet inner join AllowedPetType aPT using(AllowedPetTypeNo)
              inner join PetKind pK using(PetKindNo)
              inner join Client using(ClientNo)
    where
    Client.Login = current_user
),
--CurrentPetsReservs - знаходить номер тварини та кімнати, якщо
є дійсна наразі резервація, яку підтвердив асистент (StaffNo is
not null)
CurrentPetsReservs as
(
    select aCP.PetNo, pR.PetRoomNo, pR.StreamURL, pR.StreamType
    from AllClientPets aCP inner join Reservation r using(PetNo)
                           inner join PetRoom pR using(PetRoomNo)
    where
    r.RentStart <= current_date
    and r.RentFinish >= current_date
    and r.StaffNo is not null
)
select *
from AllClientPets aCP left outer join CurrentPetsReservs cPR
using(PetNo;
--Представлення про резервації поточного користувача-клієнта
--drop view ClientReservs;
create or replace view ClientReservs as
select
R.ReservationNo, R.PetNo, R.RentStart, R.RentFinish,
S.ServiceNo,
((SUM(S.PricePerDay) over(partition by R.ReservationNo)) +
pT.PricePerDay) * (R.RentFinish - R.RentStart) TotalPrice,
R.StaffNo, sP.Required, S.ServiceName
from
Reservation R inner join Pet using(PetNo)
              inner join AllowedPetType pT
using(AllowedPetTypeNo)
              inner join ReservService rS using(ReservationNo)
              inner join ServicePet sP using(ServicePetNo)
              inner join Service S using(ServiceNo)
              inner join Client using(ClientNo)
where Client.Login = current_user
order by R.ReservationNo;
--Представлення, про всі відгуки
--drop view AllReviews_Client;
```

Лістинг Д.1, аркуш 2

```
create or replace view AllReviews_Client as
select Rating, Comment, Date
from Review;

--Подання, тільки про працівника поточної сесії
--drop view ClientInfo;
create or replace view StaffInfo as
select name, surname, telno from Staff where Staff.login =
current_user;
-- Подання, про ті резервації, до яких прикріплений асистент або
які StaffNo = null
--drop view AllClientReservs;
create or replace view AllClientReservs as
select *
from
(
select
R.ReservationNo, R.PetNo, R.RentStart, R.RentFinish,
S.ServiceNo,
((SUM(S.PricePerDay) over(partition by R.ReservationNo)) +
pT.PricePerDay) * (R.RentFinish - R.RentStart) TotalPrice,
R.StaffNo, sP.Required, S.ServiceName
from
Reservation R inner join Pet using(PetNo)
              inner join Client using(ClientNo)
              inner join AllowedPetType pT
using(AllowedPetTypeNo)
              inner join ReservService rS using(ReservationNo)
              inner join ServicePet sP using(ServicePetNo)
              inner join Service S using(ServiceNo)
              inner join Staff using(StaffNo)
where
R.RentFinish >= current_date and Staff.Login = current_user
union
select
R.ReservationNo, R.PetNo, R.RentStart, R.RentFinish,
S.ServiceNo,
((SUM(S.PricePerDay) over(partition by R.ReservationNo)) +
pT.PricePerDay) * (R.RentFinish - R.RentStart) TotalPrice,
R.StaffNo, sP.Required, S.ServiceName
from
Reservation R inner join Pet using(PetNo)
              inner join Client using(ClientNo)
              inner join AllowedPetType pT
using(AllowedPetTypeNo)
              inner join ReservService rS using(ReservationNo)
              inner join ServicePet sP using(ServicePetNo)
              inner join Service S using(ServiceNo)
where
R.RentFinish >= current_date and R.StaffNo is null
) as myTable
```

Лістинг Д.1, аркуш 3

```
order by myTable.ReservationNo;
-- Подання, про контракти працівника поточної сесії
--drop view StaffContracts;
create or replace view StaffContracts as
select sC.StaffContractNo, Staff.login, ContractStart,
ContractFinish, PositionTitle, Salary
from
StaffContract sC inner join Staff using(StaffNo)
                 inner join Position p using(PositionNo)
where
Staff.login = current_user;
--Представлення, про ДІЙОВНІ надбавки/обчислення працівника
поточної сесії
--drop view StaffSalaryChanges;
create or replace view StaffSalaryChanges as
select sCh.SalaryChangeNo, sCh.StaffContractNo, Amount,
DateStart, DateFinish
from
SalaryChange sCh inner join StaffContract sC
using(StaffContractNo)inner join Staff using(StaffNo)
where
Staff.login = current_user and DateFinish >= current_date;
--Представлення, про вихованців, яких треба обслужити сьогодні
--drop view Pets_Groomer;
create or replace view Pets_Groomer as
select r.ReservationNo, Pet.PetNo, Pet.name, pK.KindTitle,
aPT.Size, Pet.Weight, pR.PetRoomNo
from Pet inner join AllowedPetType aPT using(AllowedPetTypeNo)
         inner join PetKind pK using(PetKindNo)
         inner join Reservation r using(PetNo)
         left outer join PetRoom pR using(PetRoomNo) --left outer
- оскільки, можливо, кімната була видалена менеджером, поки
вихованець був усе ще в готелі
where
r.RentFinish >= current_date
and r.RentStart <= current_date
and r.StaffNo is not null;
--Представлення, про кількість усіх ресурсів
--drop view ResStorage_WareHM;
create or replace view ResStorage_WareHM as
with
allResAmount as
(
    select r.ResourseNo, r.Name, 0 as amount, r.Unit
    from Resourse r
),
allResAmountPlus as
(
    --COALESCE - повертає перший не нульовий елемент зі списку
аргументів
```

Лістинг Д.1, аркуш 4

```
    --(тобто, якщо не знайдеться жодного відліку про закупівлю
такого ресурсу, то він дорівнюватиме null, що заміниться на 0)
    select distinct aRA.ResourseNo, aRA.Name,
COALESCE(SUM(pR.Amount) over(partition by ResourseNo), 0) as
amount, aRA.Unit
    from allResAmount aRA left outer join ProcurementReport pR
using(ResourseNo)
),
--select * from allResAmountPlus
allResAmountPlusMinus as
(
    select distinct aRAP.ResourseNo, aRAP.Name, (aRAP.Amount -
COALESCE(SUM(sCR.FactualAmount) over(partition by ResourseNo),
0)) as amount, aRAP.Unit
    from allResAmountPlus aRAP left outer join ServiceCostReport
sCR using(ResourseNo)
)
select Name, amount, Unit from allResAmountPlusMinus;
--Представлення, про всі ServicePet
--drop view AllServicePets_Manag;
create or replace view AllServicePets_Manag as
select s.ServiceNo, KindTitle, ServiceName, Description,
PricePerDay, Required, sP.ifActive
from
Service s inner join ServicePet sP using(ServiceNo)
          inner join PetKind pK using(PetKindNo)
where pK.ifActive = 'true';
--Представлення, про всі ServicePetResourses
--drop view AllServPetRes_Manag;
create or replace view AllServPetRes_Manag as
select KindTitle, ServiceName, res.Name, res.Unit,
sPR.AverageAmount
from
ServicePet sP inner join Service s using(ServiceNo)
              inner join PetKind pK using(PetKindNo)
              inner join ServicePetResourse sPR
using(ServicePetNo)
              inner join Resourse res using(ResourseNo);
--Представлення про всі допустимі типи тварин (ціна за день БЕЗ
урахування сервісів)
--drop view AllPetTypesInfo_Manag;
create or replace view AllPetTypesInfo_Manag as
select
KindTitle,
Size,
PricePerDay,
MaxRentDuration,
MinRentDuration,
ifActive
from
PetKind pK inner join AllowedPetType aPT using (PetKindNo);
```

Лістинг Д.1, аркуш 5

```
--Представлення, про всі кімнати
--drop view AllRooms_Manag;
create or replace view AllRooms_Manag as
select
pR.PetRoomNo,
pK.KindTitle,
aPT.Size,
pR.StreamURL,
pR.StreamType
from
PetRoom pR left outer join AllowedPetType aPT using
(AllowedPetTypeNo)
          left outer join PetKind pK using (PetKindNo);
--Представлення, про ролі ВСІХ працівників
--(ті, хто в даний момент працюють - мають певний PositionTitle)
--(ті, що вже не працюють, але зареєстровані як працівник -
мають null замість PositionTitle)
--(Якщо сьогодні останній день контратка - PositionTitle = null)
--drop view AllStaffInfo_Manag;
create or replace view AllStaffInfo_Manag as
with AllCurrentStaff_1 as
(
    select StaffNo, PositionTitle
    from StaffContract inner join Position using(PositionNo)
    where
    ContractStart <= current_date
    and
    (ContractFinish is null
    or ContractFinish > current_date)
),
AllCurrentStaff_2 as
(
    select login, Name, Surname, TelNo,
AllCurrentStaff_1.PositionTitle
    from Staff left outer join AllCurrentStaff_1 using(StaffNo)
),
AllCurrentStaff_3 as
(
    SELECT login, rolname
    FROM AllCurrentStaff_2, pg_roles
    WHERE oid IN (
        SELECT roleid
        FROM pg_auth_members
        WHERE member IN (
            SELECT oid
            FROM pg_roles
            WHERE rolname = login))
)
select *
from AllCurrentStaff_2 aCS_2 left outer join AllCurrentStaff_3
aCS_3 using(login);
```

Лістинг Д.1, аркуш 6

```
--Представлення, про контракти ВСІХ працівників
--drop view AllStaffContracts_Manag;
create or replace view AllStaffContracts_Manag as
select sC.StaffContractNo, Staff.login, ContractStart,
ContractFinish, PositionTitle, Salary
from
StaffContract sC inner join Staff using(StaffNo)
                 inner join Position p using(PositionNo);

--select * from AllStaffContracts_Manag where login = 'groom'
--Представлення, про ВСІ надбавки/обчислення ВСІХ працівників
--drop view StaffSalaryChanges_Manag;
create or replace view StaffSalaryChanges_Manag as
select sCh.SalaryChangeNo, sCh.StaffContractNo, Amount,
DateStart, DateFinish, Login
from
SalaryChange sCh inner join StaffContract sC
using(StaffContractNo)
                 inner join Staff using(StaffNo);
-1
--Для кожного клієнта визначити такі параметри:
--кількість візитів;
--кількість вихованців;
--середня кількість витрат на вихованця;
--загальну кількість витрачених клієнтом коштів;
--drop view Analitic_ClientRating;
create or replace view Analitic_ClientRating as
with Client_AllCosts as
(
    select distinct
    R.ReservationNo,
    Client.ClientNo,
    Pet.PetNo,
    COALESCE((COALESCE((SUM(S.PricePerDay) over(partition by
R.ReservationNo)), 0) + pT.PricePerDay) * (R.RentFinish -
R.RentStart), 0) TotalPrice
    from
    Reservation R inner join Pet using(PetNo)
                  inner join AllowedPetType pT
using(AllowedPetTypeNo)
                  left outer join ReservService rS
using(ReservationNo)
                  left outer join ServicePet sP
using(ServicePetNo)
                  left outer join Service S using(ServiceNo)
                  right outer join Client using(ClientNo)
    order by Client.ClientNo
)
select
Client.Name, Client.Surname,
count(distinct ReservationNo) as VisitAmount,
```

Лістинг Д.1, аркуш 7

```
count(distinct Pet.PetNo) as PetAmount,
ROUND(avg(cAllC.TotalPrice), 2) as AvgPetExpenses,
sum(cAllC.TotalPrice) as TotalPrice
from
Client inner join Pet using(ClientNo)
       --left outer - оскільки можуть бути зареєстровані тварини,
але в яких не буде контрактів
       --left outer join Reservation r using(PetNo)
       inner join Client_AllCosts cAllC using(ClientNo)
group by(Client.Name, Client.Surname, Client.ClientNo)
order by VisitAmount;
--Представлення, про всіх улюбленців, які зараз у готелі та
потребують медичного догляду
create or replace view AllReservs_Vet as
select distinct
r.ReservationNo,
p.PetNo,
p.Name,
aPT.AllowedPetTypeNo,
pK.KindTitle,
aPT.Size,
p.Weight,
p.Gender,
r.PetRoomNo,
r.MedDescription,
r.MedPrescription,
--VetStaffNo - треба, що б сортувати за призначеними на себе і
не призначеними ні на кого
r.VetStaffNo,
pR.StreamURL,
pR.StreamType
from
Reservation r inner join Pet p using (PetNo)
              inner join AllowedPetType aPT using
(AllowedPetTypeNo)
              inner join PetKind pK using (PetKindNo)
              inner join ReservService rS using (ReservationNo)
              inner join ServicePet sP using (ServicePetNo)
              inner join PetRoom pR using (PetRoomNo)
              --left outer - щоб побачити резерв. з
непризначеними на них ветеринарами
              --не використовую «using», оскільки в Reservation
два поля посилаються на Staff
              left outer join Staff S on (r.VetStaffNo =
S.StaffNo)
where
r.RentStart <= current_date
and r.RentFinish >= current_date
-- 6 - номер сервісу медичного обслуговування
and sP.ServiceNo = 6
and r.StaffNo is not null
```

Лістинг Д.1, аркуш 8

```
and ((r.VetStaffNo is null) or (S.Login = current_user));
--Представлення, про всіх PetKind
--drop view AllPetKind_Vet;
create or replace view AllPetKind_Vet as
select PetKindNo, KindTitle
from PetKind;
--Представлення, про всі AnalysisPetType
--drop view AllAnalysisPetType_Vet;
create or replace view AllAnalysisPetType_Vet as
select
AnalysisPetTypeNo,
AllowedPetTypeNo,
KindTitle,
Size,
AnalysisTypeNo,
Normative
from AnalysisPetType anPT inner join AnalysisType using
(AnalysisTypeNo)
                          inner join AllowedPetType using
(AllowedPetTypeNo)
                          inner join PetKind using (PetKindNo);
--Представлення, про всі медичні ресурси,
--які приймав вихованець
create or replace view AllMedResourses_Vet as
select
Res.Name,
sCR.FactualAmount,
Res.Unit,
sCR.ReportDate,
R.PetNo
from Resourse Res inner join ServiceCostReport sCR using
(ResourseNo)
                  inner join ReservService rS using
(ReservServiceNo)
                  inner join Reservation R using (ReservationNo)
where Res.ifMedicine = true;
```

Лістинг Д.1, аркуш 9

ДОДАТОК Е

Запити на створення користувацьких функцій бази даних

Назва функції: `ClientReg` (лістинг Е.1);

Призначення: Реєстрація нового клієнта. Перевіряє на спробу реєстрації вже існуючого номеру телефона чи логіну.

```
--Повертає:
--0 - успішна реєстрація клієнта
--1 - такий логін вже використаний
--2 - такий номер телефону вже використаний
create or replace function ClientReg(login varchar(20),
                                     password varchar(20),
                                     Name Client.Name%TYPE,
                                     Surname Client.Surname%TYPE,
                                     Telno Client.Telno%TYPE)
     returns int
as $$
declare
allLogins cursor is
    select rolname from pg_roles;
allTelNo cursor is
        select Client.telNo from Client;
begin
for currentLogin in allLogins loop
    begin
        --raise notice 'currentLogin = %; login = %;',
CAST(currentLogin AS varchar), '('|| login ||')';
        -- Додаємо дужки, оскільки приведення типів повертає
логін у дужках
        if(CAST(currentLogin AS varchar) = '('|| login ||')')
        then
            begin
                return 1;
            end;
        end if;
    end;
     end loop;
for currentTelNo in allTelNo loop
    begin
        if(currentTelNo.telNo = Telno)
        then
            begin
                return 2;
            end;
        end if;
```

Лістинг Е.1 – Запит на створення функції `ClientReg`

```
    end;
     end loop;
begin
    raise notice 'currentLogin = %; password = %;',
quote_ident(login), quote_literal(password);
    execute 'create role ' || quote_ident(login) || ' with login
password '|| quote_literal(password) ||';';
    execute 'grant client_group to '|| quote_ident(login) ||';';
    insert into Client(Name, Surname, telno, Login) values (''||
Name ||'', ''|| Surname ||'', ''|| Telno ||'', login);
     end;
return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.1, аркуш 2

Назва функції: `ReservReg` (лістинг Е.2);

Призначення: Додавання нової резервації. Перевіряє, чи є вільні кімнати, чи зарезервований обраний вихованець на інший час, чи є обраний вихованець забороненим типом тварини.

```
--Повертає:
--0 - успішна реєстрація клієнта
--1 - немає вільних кімнат
--2 - спроба зарезервувати вже зарезервованого на обраний час
улюбленця
--3 - спроба зарезервувати НЕактивного типу тварини
create or replace function ReservReg(PetNo Pet.PetNo%TYPE,
                                     RentStart
Reservation.RentStart%TYPE,
                                     RentFinish
Reservation.RentFinish%TYPE,
                                     myMedDescription
Reservation.MedDescription%TYPE,
                                     variadic ServicesNo int[])
returns int
as $$
declare
allRooms cursor is
        select distinct pR.PetRoomNo, pR.AllowedPetTypeNo
        from
        PetRoom pR
        order by pR.PetRoomNo;
allReservs cursor is
        select pR.PetRoomNo, r.RentStart, r.RentFinish, r.PetNo
```

Лістинг Е.2 – Запит на створення функції `ReservReg`

```
        from PetRoom pR inner join Reservation r using(PetRoomNo)
        where r.RentFinish >= ReservReg.RentStart;
allServicePet cursor is
        select *
        from ServicePet;
newPetTypeNo AllowedPetType.AllowedPetTypeNo%TYPE;
newPetKindNo PetKind.PetKindNo%TYPE;
emptyRoomNo PetRoom.PetRoomNo%TYPE;
ifRoomPass bool;
newReservNo Reservation.ReservationNo%TYPE;
newServicePetNo ServicePet.ServicePetNo%TYPE;
curServiceNo int;
ifMyPetKindisActive bool;
begin
    --знаходить тип тварини, який буде прописаний у резервації
    select aPT.AllowedPetTypeNo, aPT.PetKindNo into newPetTypeNo,
newPetKindNo
    from AllowedPetType aPT inner join Pet using
(AllowedPetTypeNo)
    where Pet.PetNo = ReservReg.PetNo;
    select PetKind.ifActive into ifMyPetKindisActive
    from PetKind where PetKindNo = newPetKindNo;
    if(ifMyPetKindisActive = false) then
        return 3;
    end if;
    --raise notice 'newPetTypeNo = %;', newPetTypeNo;
    -- Шукаємо вільну кімнату
    ifRoomPass = true;
    emptyRoomNo = -1;
    for curRoom in allRooms loop
    begin
        if(curRoom.AllowedPetTypeNo <> newPetTypeNo) then
continue;
        end if;
        for curReserv in allReservs loop
        begin
            --raise notice 'curRoom.PetRoomNo = %;
curReserv.PetRoomNo = %; curReserv.RentStart = %;
curReserv.RentFinish = %;', curRoom.PetRoomNo,
curReserv.PetRoomNo, curReserv.RentStart, curReserv.RentFinish;
            if(curRoom.PetRoomNo = curReserv.PetRoomNo
              and curReserv.RentStart <= ReservReg.RentFinish
              and curReserv.RentFinish >= ReservReg.RentStart)
            then
                begin
                    --Перевіряємо, що б клієнт не міг
зареєструвати одного й того самого улюбленця на дати, що
перетинаються
                    if(curReserv.PetNo = ReservReg.PetNo) then
                    begin
                        return 2;
```

Лістинг Е.2, аркуш 2

```
                    end;
                    end if;
                    --raise notice 'curRoom.AllowedPetTypeNo = %;
newPetTypeNo = %;', curRoom.AllowedPetTypeNo, newPetTypeNo;
                    ifRoomPass = false;
                    exit;
                end;
            end if;
        end;
        end loop;
        if(ifRoomPass = false)
        then
            begin
                ifRoomPass = true;
                continue;
            end;
        else
            begin
                emptyRoomNo = curRoom.PetRoomNo;
                exit;
            end;
        end if;
    end;
    end loop;
    if(emptyRoomNo = -1) then
        begin
            return 1;
        end;
    end if;
    insert into Reservation(PetNo, RentStart, RentFinish,
PetRoomNo)
    values (ReservReg.PetNo, ReservReg.RentStart,
ReservReg.RentFinish, emptyRoomNo)
    returning Reservation.ReservationNo into newReservNo;
    for curService in allServicePet loop
    begin
        if(curService.PetKindNo = newPetKindNo
          and curService.Required = true) then
        begin
            raise notice 'newReservNo = %;
curService.ServicePetNo = %;', newReservNo,
curService.ServicePetNo;
            insert into ReservService(ReservationNo,
ServicePetNo)
            values (newReservNo, curService.ServicePetNo);
        end;
        end if;
    end;
    end loop;
    raise notice 'ServicesNo = %;', array_length(ServicesNo, 1);
    --Перевіряємо, чи вибрав клієнт якісь сервіси
```

Лістинг Е.2, аркуш 3

```
    IF ServicesNo IS NULL THEN return 0;
    end if;
    --Додаємо вибрані клієнтом сервіси
    foreach curServiceNo in array ServicesNo
    loop
    begin
        for curServicePet in allServicePet loop
        begin
            if(curServicePet.ServiceNo = curServiceNo
              and curServicePet.PetKindNo = newPetKindNo) then
            begin
                insert into ReservService(ReservationNo,
ServicePetNo)
                values (newReservNo, curServicePet.ServicePetNo);
                --6 - номер медичного сервісу
                if (curServiceNo = 6) then
                begin
                    update Reservation set MedDescription =
myMedDescription
                    where ReservationNo = newReservNo;
                end;
                end if;
            end;
            end if;
        end;
        end loop;
    end;
    end loop;
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.2, аркуш 4

Назва функції: `Reserv_Cost` (лістинг Е.3);

Призначення: Обчислення ціни ще не стровеної резервації, на основі обраного типу вихованця та обраних додаткових сервісів.

```
create or replace function Reserv_Cost(PetNo Pet.PetNo%TYPE,
                                       RentStart
Reservation.RentStart%TYPE,
                                       RentFinish
Reservation.RentFinish%TYPE,
                                       variadic massServicesNo
int[])
returns numeric(9,2)
as $$
```

Лістинг Е.3 – Запит на створення функції `Reserv_Cost`

```
declare
newPetPricePerDay AllowedPetType.PricePerDay%TYPE;
newPetKindNo PetKind.PetKindNo%TYPE;
priceSummServices Service.PricePerDay%TYPE;
myResult numeric(9,2);
begin
select aPT.PricePerDay, pK.PetKindNo into newPetPricePerDay,
newPetKindNo
    from Pet inner join AllowedPetType aPT
using(AllowedPetTypeNo)
             inner join PetKind pK using(PetKindNo)
    where Pet.PetNo = Reserv_Cost.PetNo;
IF massServicesNo IS NULL THEN
    begin
        select SUM(S.PricePerDay) into priceSummServices
        from ServicePet sP inner join Service S using(ServiceNo)
        where sP.PetKindNo = newPetKindNo and sP.Required = true;
        myResult := (priceSummServices + newPetPricePerDay) *
(Reserv_Cost.RentFinish - Reserv_Cost.RentStart);
        return myResult;
    end;
else
    begin
        select SUM(S.PricePerDay) into priceSummServices
        from ServicePet sP inner join Service S using(ServiceNo)
        where sP.PetKindNo = newPetKindNo and (sP.Required = true
or S.ServiceNo = ANY(massServicesNo));
        myResult := (priceSummServices + newPetPricePerDay) *
(Reserv_Cost.RentFinish - Reserv_Cost.RentStart);
        return myResult;
    end;
end if;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.3, аркуш 2

Назва функції: `AddReview` (лістинг Е.4);

Призначення: Додає новий відгук від імені клієнта-користувача поточної сесії.

```
create or replace function AddReview(Rating Review.Rating%TYPE,
                                     Comment Review.Comment%TYPE,
                                     Date Review.Date%TYPE)
returns int
as $$
```

Лістинг Е.4 – Запит на створення функції `AddReview`

```
declare
myClientNo Client.ClientNo%TYPE;
begin
    select Client.ClientNo into myClientNo
    from
    Client
    where
    Client.Login = session_user;
    --raise notice 'myClientNo = %; current_user = %;',
myClientNo, current_user;
    insert into Review(ClientNo, Rating, Comment, Date) values
(CAST(myClientNo AS int), AddReview.Rating, AddReview.Comment,
AddReview.Date);
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.4, аркуш 2

Назва функції: `ReservRooms` (лістинг Е.5);

Призначення: Показує всі зайняті та незайняті кімнати для певного типу тварин та дати їх зайнятості.

```
create or replace function ReservRooms(newPetNo Pet.PetNo%TYPE,
                                       StartDate
Reservation.RentStart%TYPE,
                                       FinishDate
Reservation.RentFinish%TYPE)
returns table("PetRoomNo" PetRoom.PetRoomNo%TYPE,
              "RentStart" Reservation.RentStart%TYPE,
              "RentFinish" Reservation.RentFinish%TYPE)
as $$
declare
newPetTypeNo AllowedPetType.AllowedPetTypeNo%TYPE;
begin
    select Pet.AllowedPetTypeNo into newPetTypeNo
    from
    Pet
    where
    Pet.PetNo = newPetNo;
    return query
    --знаходить усі резервації для ВСІХ типів тварин дата
закінчення яких пізніше сьогоднішньої дати
    with busyReservs as
    (
        select r.PetRoomNo, RentStart, RentFinish
```

Лістинг Е.5 – Запит на створення функції `ReservRooms`

```
        from Reservation r inner join PetRoom pR using(PetRoomNo)
        where
        RentFinish >= StartDate and RentStart <= FinishDate
    )
    --Вибирає І зайняті кімнати І НЕ зайняті за ЗАДАНИМ типом
тварини
    select pR.PetRoomNo, RentStart, RentFinish
    from PetRoom pR left outer join busyReservs bR on
pR.PetRoomNo = bR.PetRoomNo
    where pR.AllowedPetTypeNo = newPetTypeNo
    order by pR.PetRoomNo, RentStart;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.5, аркуш 2

Назва функції: `ReservClientInfo_Assist` (лістинг Е.6);

Призначення: Використовується для знаходження інформації про клієнта та вихованця за номером резервації.

```
create or replace function ReservClientInfo_Assist(ReservNo
Reservation.ReservationNo%TYPE)
returns table("Login" Client.Login%TYPE,
              "Name" Client.Name%TYPE,
              "Surname" Client.Surname%TYPE,
              "Telno" Client.Telno%TYPE,
              "PetName" Pet.Name%TYPE,
              "MedDescription" Reservation.MedDescription%TYPE)
as $$
declare
begin
    return query
    select Client.Login, Client.name, Client.surname,
Client.telno, Pet.Name, R.MedDescription
    from Client inner join Pet using(ClientNo)
                inner join Reservation R using(PetNo)
    where R.ReservationNo = ReservNo;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.6 – Запит на створення функції `ReservClientInfo_Assist`

Назва функції: `ConfirmReserv_Assist` (лістинг Е.7);

Призначення: Використовується для підтвердження резервації клієнта. Тобто, коли асистент вирішує підтвердити резервацію, він, використовуючи цю функцію, прикріпляє себе к резервації, вписуючи свій табельний номер у поле `StaffNo`. Перевіряє на спробу зарезервувати неактивного типу тварини чи зарезервувати неактивні сервіси чи на спробу підтвердити вже підтверджену резервацію.

```
--Повертає:
--0 - успіх
--1 - резервація вже підтверджена або не знайдена
--2 - спроба зарезервувати НЕактивного типу тварини
--3 - спроба підтвердити резервацію з НЕактивними сервісами
create or replace function ConfirmReserv_Assist(ReservNo
Reservation.ReservationNo%TYPE)
returns int
as $$
declare
myStaffNo Staff.StaffNo%TYPE;
ifReservNull int;
ifMyPetKindisActive bool;
curServicePet RECORD;
begin
    select StaffNo into myStaffNo
    from Staff where Staff.Login = session_user;
    ifReservNull := -1;
    select COUNT(*) into ifReservNull
    from Reservation r
    where r.ReservationNo = ReservNo and r.StaffNo is null;
    if(ifReservNull = 0) then
    begin
        return 1;
    end;
    end if;
    ifMyPetKindisActive := true;
    select pK.ifActive into ifMyPetKindisActive
    from Reservation r inner join Pet using(PetNo)
                      inner join AllowedPetType aPT
using(AllowedPetTypeNo)
                      inner join PetKind pK using(PetKindNo)
    where r.ReservationNo = ReservNo;
    if(ifMyPetKindisActive = false) then
        return 2;
    end if;
    for curServicePet in
    select *
```

Лістинг Е.7 – Запит на створення функції `ConfirmReserv_Assist`

```
    from ServicePet sP inner join ReservService rS
using(ServicePetNo)
                        inner join Reservation r
using(ReservationNo)
    where r.ReservationNo = ReservNo
    loop
    begin
        raise notice 'ReservNo = %; ', curServicePet;
        if(curServicePet.ifActive = false) then
            return 3;
        end if;
    end;
    end loop;
    raise notice 'ReservNo = %; myStaffNo = %;', ReservNo,
myStaffNo;
    update Reservation set StaffNo = myStaffNo where
ReservationNo = ReservNo;
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.7, аркуш 2

Назва функції: `DelReserv_Assist` (лістинг Е.8);

Призначення: Використовується для видалення непідтверджених резервацій.

```
--Повертає:
--0 - успіх
--1 - резервація підтверджена і не може бути видалена
create or replace function DelReserv_Assist(ReservNo
Reservation.ReservationNo%TYPE)
returns int
as $$
declare
ifReservConfirm int;
begin
    ifReservConfirm := 0;
    select COUNT(*) into ifReservConfirm
    from Reservation r
    where r.ReservationNo = ReservNo and r.StaffNo is null;
    if(ifReservConfirm = 0) then
    begin
        return 1;
    end;
    end if;
```

Лістинг Е.8 – Запит на створення функції `DelReserv_Assist`

```
    delete from Reservation where ReservationNo = ReservNo;
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.8, аркуш 2

Назва функції: `PetReg_Assist` (лістинг Е.9);

Призначення: Використовується для реєстрації нового вихованця для обраного клієнта.

```
create or replace function PetReg_Assist(ClientLogin
Client.Login%TYPE,
                                        Name Pet.Name%TYPE,
                                        KindTitle
PetKind.KindTitle%TYPE,
                                        Size
AllowedPetType.Size%TYPE,
                                        Weight Pet.Weight%TYPE,
                                        Gender Pet.Gender%TYPE)
returns int

as $$
declare
myAllowedPetTypeNo AllowedPetType.AllowedPetTypeNo%TYPE;
myClientNo Client.ClientNo%TYPE;
begin
    select aPT.AllowedPetTypeNo into myAllowedPetTypeNo
    from
    PetKind pK inner join AllowedPetType aPT using(PetKindNo)
    where
    pK.KindTitle = PetReg_Assist.KindTitle
    and aPT.Size = PetReg_Assist.Size;
    select Client.ClientNo into myClientNo
    from Client where Client.Login = ClientLogin;
    --raise notice 'myClientNo = %; current_user = %;',
myClientNo, current_user;
    insert into Pet(ClientNo, Name, AllowedPetTypeNo, Weight,
Gender)
    values (CAST(myClientNo AS int), PetReg_Assist.Name,
CAST(myAllowedPetTypeNo AS int), PetReg_Assist.Weight,
PetReg_Assist.Gender);
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.9 – Запит на створення функції `PetReg_Assist`

Назва функції: `ServiceResourses_Groomer` (лістинг Е.10);

Призначення: Використовується для знаходження інформації про сервіси та ресурси для конкретної резервації, яка дійсна на момент виклику функції.

```
create or replace function ServiceResourses_Groomer(myReservNo
Reservation.ReservationNo%TYPE)
returns table("ServiceName" Service.ServiceName%TYPE,
              "Description" Service.Description%TYPE,
              "ResourseName" Resourse.Name%TYPE,
              "ResourseUnit" Resourse.Unit%TYPE,
              "AverageAmount"
ServicePetResourse.AverageAmount%TYPE)
as $$
declare
begin
    return query
    select s.ServiceName, s.Description, res.Name, res.Unit,
sPR.AverageAmount from Reservation R inner join ReservService rS
using(ReservationNo)
                inner join ServicePet sP using(ServicePetNo)
                inner join Service s using(ServiceNo)
                inner join ServicePetResourse sPR
using(ServicePetNo)inner join Resourse res using(ResourseNo)
    where r.RentFinish >= current_date
    and r.RentStart <= current_date
    and r.ReservationNo = myReservNo;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.10 – Запит на створення функції `ServiceResourses_Groomer`

Назва функції: `ServiceResourses_WareHM` (лістинг Е.11);

Призначення: Використовується для знаходження інформації про сервіси та ресурси для конкретної резервації. Вона відрізняється від попередньої функції Е.10 тим, що не має обмеження на те, чи дійсна обрана резервація.

```
create or replace function ServiceResourses_WareHM(myReservNo
Reservation.ReservationNo%TYPE)
```

Лістинг Е.11 – Запит на створення функції `ServiceResourses_WareHM`

```
returns table("ServiceName" Service.ServiceName%TYPE,
              "Description" Service.Description%TYPE,
              "ResourseName" Resourse.Name%TYPE,
              "ResourseUnit" Resourse.Unit%TYPE,
              "AverageAmount"
ServicePetResourse.AverageAmount%TYPE)
as $$
declare
begin
    return query
    select s.ServiceName,
s.Description, res.Name,
res.Unit, sPR.AverageAmount
    from Reservation R inner join ReservService rS
using(ReservationNo)
                inner join ServicePet sP using(ServicePetNo)
                inner join Service s using(ServiceNo)
                inner join ServicePetResourse sPR
using(ServicePetNo)
                inner join Resourse res using(ResourseNo)
    where
    r.ReservationNo = myReservNo;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.11, аркуш 2

Назва функції: `CostReport_WareHM` (лістинг Е.12);

Призначення: Використовується для додавання нового звіту про витрачені ресурси на обраний сервіс, обраної резервації. Перевіряє, чи достатньо обраного ресурсу на складі, та чи є кількість використаного ресурсу цілим числом, у випадку, якщо цей ресурс має приймати лише цілі значення (тобто `Unit = 'Unit'`)

```
create or replace function CostReport_WareHM(myReservNo
Reservation.ReservationNo%TYPE,
                                              myServiceName
Service.ServiceName%TYPE,
                                              myResourseName
Resourse.Name%TYPE,
                                              myAmount
ServiceCostReport.FactualAmount%TYPE,
```

Лістинг Е.12 – Запит на створення функції `CostReport_WareHM`

```
                                                    myDate
ServiceCostReport.ReportDate%TYPE)
--0 - успіх
--1 - до цілочисельного типу ресурсу намагаються присвоїти
дробову кількість
--2 - цього ресурсу не вистачає на складі
returns int
as $$
declare
myReservServiceNo ReservService.ReservServiceNo%TYPE;
myResourseNo Resourse.ResourseNo%TYPE;
myResourseUnit Resourse.Unit%TYPE;
myResStorageAmount numeric(9,2);
begin
    select ResourseNo, Unit into myResourseNo, myResourseUnit
    from Resourse where Resourse.Name = myResourseName;
    raise notice 'floor(myAmount) = %; myAmount = %;
myResourseUnit = %;', floor(myAmount), myAmount, myResourseUnit;
    if(floor(myAmount) <> myAmount
      and myResourseUnit = 'unit') then
    begin
        return 1;
    end;
    end if;
    select Amount into myResStorageAmount
    from ResStorage_WareHM where Name = myResourseName;
    if(myAmount > myResStorageAmount) then
    begin return 2;
    end;
    end if;
    select rS.ReservServiceNo into myReservServiceNo
    from
    ReservService rS inner join Reservation r
using(ReservationNo)
                    inner join ServicePet sP using(ServicePetNo)
                    inner join Service s using(ServiceNo)
                    inner join ServicePetResourse sPR
using(ServicePetNo)
    where
    r.ReservationNo = myReservNo
    and s.ServiceName = myServiceName
    and sPR.ResourseNo = myResourseNo;
    insert into
    ServiceCostReport(ReservServiceNo, ResourseNo, ReportDate,
FactualAmount)
    values
    (myReservServiceNo, myResourseNo, myDate, myAmount);
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.12, аркуш 2

Назва функції: `ProcReport_WareHM` (лістинг Е.13);

Призначення: Використовується для додавання нового звіту про доставлені ресурси. Вона, як і функція Е.12, перевіряє на те, щоб кількість ресурсу було цілим числом якщо це необхідно.

```
create or replace function ProcReport_WareHM(myResourseName
Resourse.Name%TYPE,
                                              myAmount
ServiceCostReport.FactualAmount%TYPE,
                                              myDate
ServiceCostReport.ReportDate%TYPE)
--0 - успіх
--1 - до цілочисельного типу ресурсу намагаються присвоїти
дробову кількість
returns int
as $$
declare
myResourseNo Resourse.ResourseNo%TYPE;
myResourseUnit Resourse.Unit%TYPE;
begin
    select ResourseNo, Unit into myResourseNo, myResourseUnit
    from Resourse
    where Resourse.Name = myResourseName;
    raise notice 'floor(myAmount) = %; myAmount = %;
myResourseUnit = %;', floor(myAmount), myAmount, myResourseUnit;
    if(floor(myAmount) <> myAmount
      and myResourseUnit = 'unit') then
    begin
        return 1;
    end;
    end if;
    insert into
    ProcurementReport(ResourseNo, ReportDate, Amount)
    values
    (myResourseNo, myDate, myAmount);
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.13 – Запит на створення функції `ProcReport_WareHM`

Назва функції: `AddEditTypeServ_Manag` (лістинг Е.14);

Призначення: Використовується для додавання/зміни кортежу `ServicePet`. Тобто, для призначення обраного сервісу обраному типу тварин

або редагування існуючих дозволів, змінюючи їх обов’язковість чи доступність.

```
--Повертає:
--0 - додано новий ServicePet
--1 - оновлено наявний ServicePet
create or replace function AddEditTypeServ_Manag(myServiceNo
Service.ServiceNo%TYPE,
                                                 myKindTitle
PetKind.KindTitle%TYPE,
                                                 ifRequired bool,
                                                 myifActive bool)
returns int
as $$
declare
myPetKindNo PetKind.PetKindNo%TYPE;
ifServicePetExist int;
begin
    select PetKindNo into myPetKindNo
    from PetKind
    where PetKind.KindTitle = myKindTitle;
    ifServicePetExist := 0;
    select COUNT(*) into ifServicePetExist
    from ServicePet sP
    where sP.ServiceNo = myServiceNo and sP.PetKindNo =
myPetKindNo;
    if(ifServicePetExist = 0) then
    begin
        insert into
        ServicePet(ServiceNo, PetKindNo, Required)
        values
        (myServiceNo, myPetKindNo, ifRequired);
        return 0;
    end;
    end if;
    update ServicePet sP set Required = ifRequired, ifActive =
myifActive
    where sP.ServiceNo = myServiceNo and sP.PetKindNo =
myPetKindNo;
    return 1;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.14 – Запит на створення функції `AddEditTypeServ_Manag`

Назва функції: `AddNewService_Manag` (лістинг Е.15);

Призначення: Використовується для додавання нового сервісу.

```
--Повертає:
--0 - успіх
--1 - такий ServiceName уже є
create or replace function AddNewService_Manag(myServiceName
Service.ServiceName%TYPE,
                                                myDescription
Service.Description%TYPE,
                                                myPricePerDay
Service.PricePerDay%TYPE)
returns int
as $$
declare
ifServiceNameExist int;
begin
    ifServiceNameExist := 0;
    select COUNT(*) into ifServiceNameExist
    from Service s
    where s.ServiceName = myServiceName;
    if(ifServiceNameExist <> 0) then
    begin
        return 1;
    end;
    end if;
    insert into
    Service(ServiceName, Description, PricePerDay)
    values
    (myServiceName, myDescription, myPricePerDay);
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.15 – Запит на створення функції `AddNewService_Manag`

Назва функції: `FindAvgResAmount_Manag` (лістинг Е.16);

Призначення: Використовується для знаходження середньої витрати ресурсу для обраних номеру сервісу, назви типу тварини та назви ресурсу. Повертає нуль, якщо цей ресурс не використовується для обраних параметрів.

```
create or replace function FindAvgResAmount_Manag(myServiceNo
Service.ServiceNo%TYPE,
                                                myKindTitle
PetKind.KindTitle%TYPE,
                                                myResourseName
```

Лістинг Е.16 – Запит на створення функції `FindAvgResAmount_Manag`

```
Resourse.Name%TYPE)
returns numeric(9,2)
as $$
declare
myServicePetNo int;
resultAmount ServicePetResourse.AverageAmount%TYPE;
begin
    select sP.ServicePetNo into myServicePetNo
    from ServicePet sP inner join Service s using(ServiceNo)
                       inner join PetKind pK using(PetKindNo)
    where pK.KindTitle = myKindTitle and s.ServiceNo =
myServiceNo;
    resultAmount := 0;
    select sPR.AverageAmount into resultAmount
    from ServicePetResourse sPR inner join Resourse
using(ResourseNo)
    where Resourse.Name = myResourseName and sPR.ServicePetNo =
myServicePetNo;
    if(resultAmount is null) then return 0;
    else return resultAmount;
    end if;
    return resultAmount;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.16, аркуш 2

Назва функції: `AddEditServPetRes_Manag` (лістинг Е.17);

Призначення: Використовується для додавання/зміни кортежу `ServicePetResourse`. Тобто, для призначення кількості використання обраного ресурсу для обраного сервісу та типу тварини або для редагування існуючих призначень, змінюючи кількість ресурсу, який використовується.

```
create or replace function AddEditServPetRes_Manag(myServiceNo
Service.ServiceNo%TYPE,
                                                    myKindTitle
PetKind.KindTitle%TYPE,
                                                    myResourseName
Resourse.Name%TYPE,
                                                    myAvgAmount
ServicePetResourse.AverageAmount%TYPE)
--Повертає:
--0 - додано новий ServicePetResourse
--1 - оновлено наявний ServicePetResourse
--2 - обраний ресурс = 'unit', а myAvgAmount не цілочисельне
```

Лістинг Е.17 – Запит на створення функції `AddEditServPetRes_Manag`

```
returns int
as $$
declare
ifSPRexist int;
myServicePetNo ServicePet.ServicePetNo%TYPE;
myResourseNo Resourse.ResourseNo%TYPE;
myResourseUnit Resourse.Unit%TYPE;
begin
    ifSPRexist := 0;
    select COUNT(*) into ifSPRexist
    from
    ServicePet sP inner join Service s using(ServiceNo)
                  inner join PetKind pK using(PetKindNo)
                  inner join ServicePetResourse sPR
using(ServicePetNo)
                  inner join Resourse res using(ResourseNo)
    where
    s.ServiceNo = myServiceNo
    and pK.KindTitle = myKindTitle
    and res.Name = myResourseName;
    select sP.ServicePetNo into myServicePetNo
    from
    ServicePet sP inner join Service s using(ServiceNo)
                  inner join PetKind pK using(PetKindNo)
    where
    s.ServiceNo = myServiceNo
    and pK.KindTitle = myKindTitle;
    select ResourseNo, Unit into myResourseNo, myResourseUnit
    from Resourse where Name = myResourseName;
    if(floor(myAvgAmount) <> myAvgAmount
      and myResourseUnit = 'unit') then
    begin
        return 2;
    end;
    end if;
    if(ifSPRexist = 0) then
    begin
        insert into
        ServicePetResourse(ServicePetNo, ResourseNo,
AverageAmount)
        values(myServicePetNo, myResourseNo, myAvgAmount);
        return 0;
    end;
    end if;
    update ServicePetResourse sPR set AverageAmount = myAvgAmount
    where sPR.ServicePetNo = myServicePetNo and sPR.ResourseNo =
myResourseNo;
    return 1;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.17, аркуш 2

Назва функції: DeleteServPetRes_Manag (лістинг Е.18);

Призначення: Використовується для видалення кортежу ServicePetResourse. Тобто, для скасування використання ресурсу для обраного сервісу і типу тварини.

```
--Повертає:
--0 - успіх
--1 - такого ServicePetResourse немає
create or replace function DeleteServPetRes_Manag(myServiceNo
Service.ServiceNo%TYPE,
                                                  myKindTitle
PetKind.KindTitle%TYPE,
                                                  myResourseName
Resourse.Name%TYPE)
returns int
as $$
declare
ifSPRexist int;
myServicePetNo ServicePet.ServicePetNo%TYPE;
myResourseNo Resourse.ResourseNo%TYPE;
begin
    ifSPRexist := 0;
    select COUNT(*) into ifSPRexist
    from
    ServicePet sP inner join Service s using(ServiceNo)
                  inner join PetKind pK using(PetKindNo)
                  inner join ServicePetResourse sPR
using(ServicePetNo)
                  inner join Resourse res using(ResourseNo)
    where
    s.ServiceNo = myServiceNo
    and pK.KindTitle = myKindTitle
    and res.Name = myResourseName;
    if(ifSPRexist = 0) then
    begin
        return 1;
    end;
    end if;
    select sP.ServicePetNo into myServicePetNo
    from
    ServicePet sP inner join Service s using(ServiceNo)
                  inner join PetKind pK using(PetKindNo)
    where
    s.ServiceNo = myServiceNo
    and pK.KindTitle = myKindTitle;
    select ResourseNo into myResourseNo
    from Resourse where Name = myResourseName;
    delete from ServicePetResourse sPR
```

Лістинг Е.18 – Запит на створення функції DeleteServPetRes_Manag

```
    where sPR.ServicePetNo = myServicePetNo and sPR.ResourseNo =
myResourseNo;
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.18, аркуш 2

Назва функції: `AddNewResourse_Manag` (лістинг Е.19);

Призначення: Використовується для додавання нового ресурсу.

```
--Повертає:
--0 - успіх
--1 - такий ResourseName вже є
create or replace function AddNewResourse_Manag(myResourseName
Resourse.Name%TYPE,
                                                myResourseUnit
Resourse.Unit%TYPE,
                                                myIfMedicine
Resourse.ifMedicine%TYPE)
returns int
as $$
declare
ifResourseExist int;
begin
    ifResourseExist := 0;
    select COUNT(*) into ifResourseExist
    from Resourse res
    where res.Name = myResourseName;
    if(ifResourseExist <> 0) then
    begin
        return 1;
    end;
    end if;
    insert into
    Resourse(Name, Unit, ifMedicine)
    values
    (myResourseName, myResourseUnit, myIfMedicine);
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.19 – Запит на створення функції `AddNewResourse_Manag`

Назва функції: `EditPetType` (лістинг Е.20);

Призначення: Використовується для редагування інформації про таблиці `PetKind` та `AllowedPetType`. Тобто, для редагування: назви типу тварин, максимальний та мінімальний термін резервації, ціна за день для кожного з трьох розмірів обраного типу тварини та доступності цього типу тварин.

```
create or replace function EditPetType(oldKindTitle
PetKind.KindTitle%TYPE,
                                       myKindTitle
PetKind.KindTitle%TYPE,
                                       myMaxRentDur
PetKind.MaxRentDuration%TYPE,
                                       myMinRentDur
PetKind.MinRentDuration%TYPE,
                                       myPricePerDay
numeric(9,2)[],
                                       myIfActive bool)
returns int
as $$
declare
myPetKindNo PetKind.PetKindNo%TYPE;
begin
    select PetKindNo into myPetKindNo
    from PetKind where KindTitle = oldKindTitle;
    update PetKind set
    KindTitle = myKindTitle,
    MaxRentDuration = myMaxRentDur,
    MinRentDuration = myMinRentDur,
    ifActive = myIfActive
    where PetKindNo = myPetKindNo;
    raise notice 'myPricePerDay[0] = %;', myPricePerDay[0];
    update AllowedPetType set
    PricePerDay = myPricePerDay[1]
    where PetKindNo = myPetKindNo and Size = 'Small';
    update AllowedPetType set
    PricePerDay = myPricePerDay[2]
    where PetKindNo = myPetKindNo and Size = 'Medium';
    update AllowedPetType set
    PricePerDay = myPricePerDay[3]
    where PetKindNo = myPetKindNo and Size = 'Big';
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.20 – Запит на створення функції `EditPetType`

Назва функції: `AddNewPetType` (лістинг Е.21);

Призначення: Використовується для додавання нових записів у таблиці `PetKind` та `AllowedPetType`. Тобто, для додавання нового типу тварин, призначаючи їм максимальний та мінімальний термін резервації та ціну за день для кожного з трьох розмірів нового типу тварини.

```
create or replace function AddNewPetType(myKindTitle
PetKind.KindTitle%TYPE,
                                        myMaxRentDur
PetKind.MaxRentDuration%TYPE,
                                        myMinRentDur
PetKind.MinRentDuration%TYPE,
                                        myPricePerDay
numeric(9,2)[])
returns int
as $$
declare
myPetKindNo PetKind.PetKindNo%TYPE;
begin
    insert into PetKind(KindTitle, MaxRentDuration,
MinRentDuration)
    values (myKindTitle, myMaxRentDur, myMinRentDur)
    returning PetKind.PetKindNo into myPetKindNo;
    insert into  AllowedPetType(PetKindNo, Size, PricePerDay)
    values myPetKindNo, 'Small', myPricePerDay[1]),
    (myPetKindNo, 'Medium', myPricePerDay[2]),
    (myPetKindNo, 'Big', myPricePerDay[3]);
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.21 – Запит на створення функції `AddNewPetType`

Назва функції: `AddNewRoom` (лістинг Е.22);

Призначення: Використовується для додавання нової кімнати для обраного типу тварини.

```
create or replace function AddNewRoom(myKindTitle
PetKind.KindTitle%TYPE, myPetSize AllowedPetType.Size%TYPE,
myStreamURL PetRoom.StreamURL%TYPE, myStreamType
```

Лістинг Е.22 – Запит на створення функції `AddNewRoom`

```
 PetRoom.StreamType%TYPE)
returns int
as $$
declare
myAPTNo AllowedPetType.AllowedPetTypeNo%TYPE;
begin
    select aPT.AllowedPetTypeNo into myAPTNo
    from AllowedPetType aPT inner join PetKind pK
using(PetKindNo)
    where aPT.Size = myPetSize and pK.KindTitle = myKindTitle;
    insert into
    PetRoom(AllowedPetTypeNo, StreamURL, StreamType)
    values
    (myAPTNo, myStreamURL, myStreamType);
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.22, аркуш 2

Назва функції: `AddSalChange_Manag` (лістинг Е.23);

Призначення: Використовується для додавання нової надбавки чи вирахування для обраного трудового договору. Перевіряє, чи закінчився цей договір, та чи не перевищує сума вирахування зарплатню працівника.

```
--Повертає:
--0 - успіх
--1 - StaffContract уже закінчився
--2 - Відрахування більше ніж зарплата посади у цього контракту
create or replace function AddSalChange_Manag(myStaffContractNo
StaffContract.StaffContractNo%TYPE,
                                              myAmount
SalaryChange.Amount%TYPE,
                                              myDateStart
SalaryChange.DateStart%TYPE,
                                              myDateFinish
SalaryChange.DateFinish%TYPE)
returns int
as $$
declare
myContractFinish StaffContract.ContractFinish%TYPE;
mySalary Position.Salary%TYPE;
begin
```

Лістинг Е.23 – Запит на створення функції `AddSalChange_Manag`

```
    select ContractFinish, p.Salary into myContractFinish,
mySalary
    from StaffContract inner join Position p using(PositionNo)
    where StaffContractNo = myStaffContractNo;
    if(myContractFinish < current_date) then
    begin
        return 1;
    end;
    end if;
    if(mySalary + myAmount < 0) then
    begin
        return 2;
    end;
    end if;
    insert into SalaryChange(StaffContractNo, Amount, DateStart,
DateFinish)
    values(myStaffContractNo, myAmount, myDateStart,
myDateFinish);
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.23, аркуш 2

Назва функції: `AddStaffContract_Manag` (лістинг Е.24);

Призначення: Використовується для додавання нового трудового договору для обраного працівника на обрані посаду та термін дії договору. Перевіряє, чи є у працівника договори, які перетинаються з новим договором.

```
--Повертає:
--0 - успіх
--1 - у обраного працівника є незавершені безстрокові контракти
(StaffContract.ContractFinish = null)
--2 - обрані дати перетинаються з іншим контрактом
create or replace function AddStaffContract_Manag(myStaffLogin
Staff.Login%TYPE,
                                                  myPositionTitle
varchar,
                                                  myContractStart
StaffContract.ContractStart%TYPE,
                                                  myContractFinish
StaffContract.ContractFinish%TYPE)
returns int
as $$
declare
```

Лістинг Е.24 – Запит на створення функції `AddStaffContract_Manag`

```
myStaffNo Staff.StaffNo%TYPE;
myPositionNo Position.PositionNo%TYPE;
curStaffContract RECORD;
begin
    select StaffNo into myStaffNo
    from Staff
    where Login = myStaffLogin;
    select PositionNo into myPositionNo
    from Position
    where PositionTitle = myPositionTitle;
    for curStaffContract in
    select * from StaffContract sC where sC.StaffNo = myStaffNo
    loop
    begin
        if(curStaffContract.ContractFinish is null
          and curStaffContract.ContractStart <= myContractFinish)
then
        begin
            return 1;
        end;
        end if;
        if((curStaffContract.ContractStart <= myContractFinish
           or myContractFinish is null)
           and curStaffContract.ContractFinish >=
myContractStart) then
        begin
            return 2;
        end;
        end if;
    end;
    end loop;
    insert into
    StaffContract(StaffNo, PositionNo, ContractStart,
ContractFinish)
    values
    (myStaffNo, myPositionNo, myContractStart, myContractFinish);
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.24, аркуш 2

Назва функції: `GiveRole_Manag` (лістинг Е.25);

Призначення: Використовується для видачі прав на обрану посаду для обраного працівника. Перевіряє, щоб працівник не міг мати права одразу на кілька посад.

```
--Повертає:
--0 - успіх
--1 - Staff уже має роль
--2 - Не знайшлося такого myPosTitle
create or replace function GiveRole_Manag(myLogin
Staff.Login%TYPE,
                                          myPosTitle varchar)
returns int
as $$
declare
ifStaffHasRole int;
begin
    ifStaffHasRole := 0;
    SELECT Count(rolname) into ifStaffHasRole
    FROM pg_roles
    WHERE oid IN (
        SELECT roleid
        FROM pg_auth_members
        WHERE member IN (
            SELECT oid
            FROM pg_roles
            WHERE rolname = myLogin
        )
    );
    if(ifStaffHasRole <> 0) then
    begin
        return 1;
    end;
    end if;
    if(myPosTitle = 'Assistant') then
    begin
        execute format('grant assist_group to %I;', myLogin);
        return 0;
    end;
    end if;
    if(myPosTitle = 'Groomer') then
    begin
        execute format('grant groomer_group to %I;', myLogin);
        return 0;
    end;
    end if;
    if(myPosTitle = 'Warehouse manager') then
    begin
        execute format('grant warehm_group to %I;', myLogin);
        return 0;
    end;
    end if;
    return 2;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.25 – Запит на створення функції `GiveRole_Manag`

Назва функції: `TakeRole_Manag` (лістинг Е.26);

Призначення: Використовується для того, щоб відібрати права на обрану посаду від обраного працівника. Перевіряє наявність прав у працівника, які планується видалити.

```
--Повертає:
--0 - успіх
--1 - у співробітника немає прав для видалення
create or replace function TakeRole_Manag(myLogin
Staff.Login%TYPE)
returns int
as $$
declare
ifStaffHasRole varchar;
begin
    ifStaffHasRole := null;
    SELECT rolname into ifStaffHasRole
    FROM pg_roles
    WHERE oid IN (
        SELECT roleid
        FROM pg_auth_members
        WHERE member IN (
            SELECT oid
            FROM pg_roles
            WHERE rolname = myLogin
        )
    );
    if(ifStaffHasRole is null) then
    begin
        return 1;
    end;
    end if;
    execute format('revoke %I from %I', ifStaffHasRole, myLogin);
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.26 – Запит на створення функції `TakeRole_Manag`

Назва функції: `StaffReg_Manag` (лістинг Е.27);

Призначення: Використовується для реєстрації нового працівника. Перевіряє, чи використовується вже обраний номер телефону чи логін нового працівника.

```
--Повертає:
--0 - успішна реєстрація співробітника
--1 - такий логін уже використано
--2 - такий номер телефону вже використано
create or replace function StaffReg_Manag(login varchar(20),
                                          password varchar(20),
                                          Name Client.Name%TYPE,
                                          Surname Client.Surname%TYPE,
                                          Telno Client.Telno%TYPE)
returns int
as $$
declare
allLogins cursor is select rolname from pg_roles;
allTelNo cursor is select Staff.telNo from Staff;
begin
for currentLogin in allLogins loop
    begin
        --raise notice 'currentLogin = %; login = %;',
CAST(currentLogin AS varchar), '('|| login ||')';
        -- Додаємо дужки, оскільки приведення типів повертає
логін у дужках
        if(CAST(currentLogin AS varchar) = '('|| login ||')')
        then
            begin
                return 1;
            end;
        end if;
    end;
end loop;
for currentTelNo in allTelNo loop
    begin
        if(currentTelNo.telNo = Telno)
        then
            begin
                return 2;
            end;
        end if;
    end;
end loop;
begin
    execute 'create role ' || quote_ident(login) || ' with login
password '|| quote_literal(password) ||';';
    insert into Staff(Name, Surname, telno, Login) values (''||
Name ||'', ''|| Surname ||'', ''|| Telno ||'', login);
end;
return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.27 – Запит на створення функції `StaffReg_Manag`

Назва функції: `Analitic_MonthProfit` (лістинг Е.28);

Призначення: Використовується для обчислення прибутку за всі місяці обраного року.

```
create or replace function Analitic_MonthProfit(myYear int)
returns table("MonthNo" numeric,"MonthProfit" numeric(9,2))
as $$
declare
begin
    return query
    with
    allR as
    (
        select distinct
        R.ReservationNo,
        R.RentStart as StartDate,
        COALESCE((COALESCE((SUM(S.PricePerDay) over(partition by
R.ReservationNo)), 0) + pT.PricePerDay) * (R.RentFinish -
R.RentStart), 0) TotalPrice
        from
        Reservation R inner join Pet using(PetNo)
                      inner join AllowedPetType pT
using(AllowedPetTypeNo)
                      left outer join ReservService rS
using(ReservationNo)
                      left outer join ServicePet sP
using(ServicePetNo)
                      left outer join Service S using(ServiceNo)
        where extract(year from r.Rentstart) = myYear
        order by R.ReservationNo
    ),
    mP as
    (
        select
        SUM(allR.TotalPrice) as MonthProfit,
        extract (month from allR.StartDate) as MonthNo
        from allR
        group by(extract (month from allR.StartDate))
    ),
    mNumbers as
    (
        select generate_series(1, 12) as MonthNo
    )
    select MonthNo, COALESCE(MonthProfit, 0) as MonthProfit
    from mP right outer join mNumbers using(MonthNo);
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.28 – Запит на створення функції `Analitic_MonthProfit`

Назва функції: `Analitic_PopularServices` (лістинг Е.29);

Призначення: Використовується для відображення популярності (кількості резервацій, які обрали цей сервіс) кожного сервісу у вказаний проміжок часу.

```
create or replace function Analitic_PopularServices(dStart
Reservation.RentStart%TYPE,
                                                    dFinish
Reservation.RentFinish%TYPE)
returns table("ServiceName" Service.ServiceName%TYPE,
              "KindTitle" PetKind.KindTitle%TYPE,
              "Description" Service.Description%TYPE,
              "PricePerDay" Service.PricePerDay%TYPE,
              "Required" ServicePet.Required%TYPE,
              "Amount" bigint)
as $$
declare
begin
    return query
    select distinct
    s.ServiceName,
    pK.KindTitle,
    s.Description,
    s.PricePerDay,
    sP.Required,
    count(s.ServiceName) over(partition by sP.ServicePetNo)
    from Reservation r inner join ReservService rS
using(ReservationNo)
                       inner join ServicePet sP
using(ServicePetNo)
                       right outer join Service s
using(ServiceNo)
                       right outer join PetKind pK
using(PetKindNo)
    where r.RentStart <= dFinish and r.RentFinish >= dStart;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.29 – Запит на створення функції `Analitic_PopularServices`

Назва функції: `Analitic_ServCostReport` (лістинг Е.30);

Призначення: Використовується для відображення інформації про витрати всіх типів ресурсу на всі типи тварин за певний проміжок часу.

```
create or replace function Analitic_ServCostReport(dStart
Reservation.RentStart%TYPE, dFinish Reservation.RentFinish%TYPE)
returns table("ServiceName" Service.ServiceName%TYPE,
              "KindTitle" PetKind.KindTitle%TYPE,
              "Name" Resourse.Name%TYPE,
              "Unit" Resourse.Unit%TYPE,
              "FactualAmount" numeric,
              "AverageAmount" numeric)
as $$
declare
begin
    return query
    select s.ServiceName, pK.KindTitle, res.Name, res.Unit,
    sum(sCR.FactualAmount),
    sPR.AverageAmount * count(sCR.FactualAmount)
    from ServiceCostReport sCR inner join ReservService rS
using(ReservServiceNo)
                                inner join ServicePet sP
using(ServicePetNo)
                                inner join Service s
using(ServiceNo)
                                inner join PetKind pK
using(PetKindNo)
                                inner join Resourse res
using(ResourseNo)
                                inner join ServicePetResourse sPR
using(ServicePetNo)
    where sCR.ReportDate <= dFinish and sCR.ReportDate >= dStart
    group by (s.ServiceName,
pK.KindTitle,res.Name,res.Unit,sPR.AverageAmount);
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.30 – Запит на створення функції `Analitic_ServCostReport`

Назва функції: `Analitic_StaffReserv` (лістинг Е.31);

Призначення: Використовується для відображення ефективності роботи асистентів. Функція відображає кожного асистента, кількість прийнятих ним контрактів та їх загальну вартість.

```
create or replace function Analitic_StaffReserv(dStart
Reservation.RentStart%TYPE,
                                                 dFinish
Reservation.RentFinish%TYPE)
```

Лістинг Е.31 – Запит на створення функції `Analitic_StaffReserv`

```
returns table("Name" Staff.Name%TYPE,
              "Surname" Staff.Surname%TYPE,
              "TelNo" Staff.TelNo%TYPE,
              "PriceSum" numeric,
              "Amount" bigint)
as $$
declare
begin
    return query
    with allStaffReserv as
    (
        select distinct
        Staff.StaffNo,
        COALESCE((COALESCE((SUM(S.PricePerDay) over(partition by
R.ReservationNo)), 0) + pT.PricePerDay) * (R.RentFinish -
R.RentStart), 0) TotalPrice
        from
        Reservation R inner join Pet using(PetNo)
                      inner join AllowedPetType pT
using(AllowedPetTypeNo)
                      left outer join ReservService rS
using(ReservationNo)
                      left outer join ServicePet sP
using(ServicePetNo)
                      left outer join Service S using(ServiceNo)
                      inner join Staff using(StaffNo)
        where r.RentStart <= dFinish and r.RentFinish >= dStart
    )
    select Staff.Name, Staff.Surname, Staff.TelNo,
    sum(TotalPrice),
    count(StaffNo)
    from allStaffReserv inner join Staff using(StaffNo)
    group by(Staff.Name, Staff.Surname, Staff.TelNo);
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.31, аркуш 2

Назва функції: `AssignRreserv_Vet` (лістинг Е.32);

Призначення: Використовується для призначення поточного користувача (ветеринара) на обрану резервацію. Функція перевіряє, чи не призначений вже цей вихованець на цього або на іншого ветеринара.

```
create or replace function AssignRreserv_Vet(myReservationNo
Reservation.ReservationNo%TYPE)
```

Лістинг Е.32 – Запит на створення функції `AssignRreserv_Vet`

```
returns int
as $$
declare
myVetStaffNo Staff.StaffNo%TYPE;
oldVetStaffNo Reservation.VetStaffNo%TYPE;
begin
    select StaffNo into myVetStaffNo
    from Staff
    where Login = session_user;
    select VetStaffNo from Reservation where ReservationNo =
myReservationNo into oldVetStaffNo;
    if (oldVetStaffNo = myVetStaffNo) then
        RAISE EXCEPTION 'Цей вихованець вже призначений на вас!'
        USING ERRCODE = 'P0001';
    end if;
    --(is distinct from) краще <>, оскільки за <>: null <> null
    if (oldVetStaffNo is distinct from NULL) then
        RAISE EXCEPTION 'Цей вихованець вже призначений на іншого
ветеринара!'
        USING ERRCODE = 'P0002';
    end if;
    update Reservation set VetStaffNo = myVetStaffNo
    where ReservationNo = myReservationNo;
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.32, аркуш 2

Назва функції: `EditMedPrescription_Vet` (лістинг Е.33);

Призначення: Використовується для зміни призначеного лікування для вихованця. Функція перевіряє, чи є цей вихованець призначеним на поточного ветеринара (ветеринар не має право змінювати назначені ліки вихованців, які не призначені на нього).

```
create or replace function
EditMedPrescription_Vet(myReservationNo
Reservation.ReservationNo%TYPE,
                                                    myMedPrescription
Reservation.MedPrescription%TYPE)
returns int
as $$
declare
myVetStaffNo Staff.StaffNo%TYPE;
```

Лістинг Е.33 – Запит на створення функції `EditMedPrescription_Vet`

```
oldVetStaffNo Reservation.VetStaffNo%TYPE;
begin
    select StaffNo into myVetStaffNo
    from Staff
    where Login = session_user;
    select VetStaffNo from Reservation where ReservationNo =
myReservationNo into oldVetStaffNo;
    if (oldVetStaffNo is distinct from myVetStaffNo) then
        RAISE EXCEPTION 'Цей вихованець не призначений на вас!'
        USING ERRCODE = 'P0001';
    end if;
    update Reservation set MedPrescription = myMedPrescription
    where ReservationNo = myReservationNo;
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.33, аркуш 2

Назва функції: `AddAnalysisType_Vet` (лістинг Е.34);

Призначення: Використовується для додавання нового типу аналізу. Функція необхідна, аби не надавати ветеринарам зайвих прав на внутрішню послідовність таблиці `AnalysisType`.

```
create or replace function AddAnalysisType_Vet(myName
AnalysisType.Name%TYPE)
returns int
as $$
begin
    insert into AnalysisType(Name) values (myName);
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.34 – Запит на створення функції `AddAnalysisType_Vet`

Назва функції: `AddAnalysisPetType_Vet` (лістинг Е.35);

Призначення: Використовується для додавання нового відношення типу аналізу та типу тварина. Функція додає нове нормативне значення обраного типу аналізу для обраного типу тварини.

```
create or replace function
EditMedPrescription_Vet(myReservationNo
Reservation.ReservationNo%TYPE,
                                                    myMedPrescription
Reservation.MedPrescription%TYPE)
returns int
as $$
declare
myVetStaffNo Staff.StaffNo%TYPE;
oldVetStaffNo Reservation.VetStaffNo%TYPE;
begin
    select StaffNo into myVetStaffNo
    from Staff
    where Login = session_user;

    select VetStaffNo from Reservation where ReservationNo =
myReservationNo into oldVetStaffNo;
    if (oldVetStaffNo is distinct from myVetStaffNo) then
        RAISE EXCEPTION 'Цей вихованець не призначений на вас!'
        USING ERRCODE = 'P0001';
    end if;
    update Reservation set MedPrescription = myMedPrescription
    where ReservationNo = myReservationNo;
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.35 – Запит на створення функції `AddAnalysisPetType_Vet`

Назва функції: `AddAnalysis_Vet` (лістинг Е.36);

Призначення: Використовується для додавання нового результату аналізу для обраного вихованця. Функція перевіряє, чи є обраний вихованець призначений на поточного ветеринара (ветеринари можуть додавати результати аналізів тільки для тих тварин, які є призначені на них).

```
create or replace function AddAnalysis_Vet(myPetNo
Pet.PetNo%TYPE,
                                          myAnalysisPetTypeNo
AnalysisPetType.AnalysisPetTypeNo%TYPE,
                                          myResult
Analysis.Result%TYPE)
returns int
as $$
declare
```

Лістинг Е.36 – Запит на створення функції `AddAnalysis_Vet`

```
myVetStaffNo Staff.StaffNo%TYPE;
currentVetStaffNo Reservation.VetStaffNo%TYPE;
myReservationNo Reservation.ReservationNo%TYPE;
begin
    select StaffNo into myVetStaffNo
    from Staff
    where Login = session_user;
    select R.ReservationNo into myReservationNo
    from Reservation R
    where
    R.PetNo = myPetNo
    and R.RentStart <= current_date
    and R.RentFinish >= current_date;
    select VetStaffNo from Reservation where ReservationNo =
myReservationNo into currentVetStaffNo;
    if (currentVetStaffNo is distinct from myVetStaffNo) then
        RAISE EXCEPTION 'Цей вихованець не призначений на вас!'
        USING ERRCODE = 'P0001';
    end if;
    insert into
    Analysis(PetNo, AnalysisPetTypeNo, Result)
    values
    (myPetNo, myAnalysisPetTypeNo, myResult);
    return 0;
end;
$$ language plpgsql
SECURITY DEFINER;
```

Лістинг Е.36, аркуш 2

ДОДАТОК Ж

Привілеї ролей на об'єкти БД

Таблиця Ж.1 – Привілеї ролей на об'єкти БД

Об'єкти БД	Ролі					
	Клієнт `client_group`	Асистент `assist_group`	Грумер `groomer_group`	Завскладом `warehm_group`	Менеджер `manag_group`	Ветеринар `vet_group`
Таблиці						
`Client`		R				
`Review`						
`PetKind`		R			RU	
`AllowedPetType`		R			R	
`PetRoom`					RUD	
`Pet`		R				
`Staff`						
`Position`					RU	
`StaffContract`		R			RUD	
`SalaryChange`		R			RUD	
`Reservation`						

Продовження таблиці Ж.1

Об'єкти БД	Ролі					
	Клієнт `client_group`	Асистент `assist_group`	Грумер `groomer_group`	Завскладом `warehm_group`	Менеджер `manag_group`	Ветеринар `vet_group`
`Service`					RU	
`ServicePet`					RU	
`Resourse`				R	RUD	
`ServicePetResourse`						
`ReservService`						
`ProcurementReport`						
`ServiceCostReport`						
`Analysis`						R
`AnalysisPetType`						RUD
`AnalysisType`						RUD
Представлення						
`ClientInfo`	RU					
`AllServicePets`	R	R				
`AllPetTypesInfo`	R	R				

Продовження таблиці Ж.1

Об'єкти БД	Ролі					
	Клієнт `client_group`	Асистент `assist_group`	Грумер `groomer_group`	Завскладом `warehm_group`	Менеджер `manag_group`	Ветеринар `vet_group`
`ClientPets`	R					
`ClientReservs`	R					
`AllReviews_Client`	R					
`AllClientReservs`		R				
`StaffSalaryChanges`		R	R	R		R
`StaffContracts`		R	R	R		R
`StaffInfo`		RU	RU	RU		RU
`Pets_groomer`			R			
`ResStorage_WareHM`				R		
`AllServicePets_Manag`					R	
`AllServPetRes_Manag`					R	
`AllPetTypesInfo_Manag`					R	
`AllRooms_Manag`					R	

Продовження таблиці Ж.1

Об'єкти БД	Ролі					
	Клієнт client_group	Асистент assist_group	Грумер groomer_group	Завскладом warehm_group	Менеджер manag_group	Ветеринар vet_group
AllStaffInfo_Manag					R	
AllStaffContracts_Manag					R	
StaffSalaryChanges_Manag					R	
Analitic_ClientRating					R	
Analitic_ProcReports					R	
Analitic_ClientComments					R	
AllReservs_Vet						R
AllPetKind_Vet						R
AllAnalysisPetType_Vet						R
AllMedResourses_Vet						R
Функції						
PetReg	E					
ReservReg	E					

Продовження таблиці Ж.1

Об'єкти БД	Ролі					
	Клієнт `client_group`	Асистент `assist_group`	Грумер `groomer_group`	Завскладом `warehm_group`	Менеджер `manag_group`	Ветеринар `vet_group`
`Reserv_Cost`	E					
`AddReview`	E					
`ReservRooms`	E					
`ReservClientInfo_Assist`		E				
`ConfirmReserv_Assist`		E				
`DelReserv_Assist`		E				
`PetReg_Assist`		E				
`ClientReg`		E				
`ServiceResourses_Groomer`			E			
`ServiceResourses_WareHM`				E		
`CostReport_WareHM`				E		
`ProcReport_WareHM`				E		
`AddEditTypeServ_Manag`					E	

Продовження таблиці Ж.1

Об'єкти БД	Ролі					
	Клієнт client_group	Асистент assist_group	Грумер groomer_group	Завскладом warehm_group	Менеджер manag_group	Ветеринар vet_group
AddNewService_Manag					E	
FindAvgResAmount_Manag					E	
AddEditServPetRes_Manag					E	
DeleteServPetRes_Manag					E	
EditPetType					E	
AddNewPetType					E	
AddNewRoom					E	
AddSalChange_Manag					E	
GiveRole_Manag					E	
TakeRole_Manag					E	
StaffReg_Manag					E	
Analitic_MonthProfit					E	
Analitic_PopularServices					E	

Продовження таблиці Ж.1

Об'єкти БД	Ролі					
	Клієнт client_group	Асистент assist_group	Грумер groomer_group	Завскладом warehm_group	Менеджер manag_group	Ветеринар vet_group
Analitic_ServCostReport					E	
Analitic_StaffReserv					E	
AssignRreserv_Vet						E
EditMedPrescription_Vet						E
AddAnalysisType_Vet						E
AddAnalysisPetType_Vet						E
AddAnalysis_Vet						E

ДОДАТОК И

Створення ролей і наділення їх привілеями

```
--Клієнт--
--Створення групи--
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM
client_group;
drop role client_group;
create role client_group NOLOGIN;
--Таблиці та представлення--
grant select on ClientInfo to client_group;
grant update on ClientInfo to client_group;
grant select on AllPetTypesInfo to client_group;
grant select on AllServicePets to client_group;
grant select on ClientPets to client_group;
grant select on ClientReservs to client_group;
grant select on AllReviews_Client to client_group;
--Функції--
grant execute on function PetReg to client_group;
grant execute on function ReservReg to client_group;
grant execute on function Reserv_Cost to client_group;
grant execute on function AddReview to client_group;
grant execute on function ReservRooms to client_group;
--Асистент--
--Створення групи--
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM
assist_group;
drop role assist_group;
create role assist_group NOLOGIN;
--Таблиці та представлення--
grant select on StaffInfo to assist_group;
grant update on StaffInfo to assist_group;
grant select on AllPetTypesInfo to assist_group;
grant select on AllServicePets to assist_group;
grant select on AllClientReservs to assist_group;
grant select on Client to assist_group;
grant select on Pet to assist_group;
grant select on AllowedPetType to assist_group;
grant select on PetKind to assist_group;
grant select on StaffContracts to assist_group;
grant select on StaffSalaryChanges to assist_group;
--Функції--
grant execute on function ReservClientInfo_Assist to
assist_group;
grant execute on function ConfirmReserv_Assist to assist_group;
grant execute on function DelReserv_Assist to assist_group;
grant execute on function PetReg_Assist to assist_group;
grant execute on function ClientReg to assist_group;
```

Лістинг И.1 – Запити на створення ролей і наділення їх привілеями

```
--Грумер--
--Створення групи--
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM
groomer_group;
drop role groomer_group;
create role groomer_group NOLOGIN;
--Таблиці та представлення--
grant select on StaffInfo to groomer_group;
grant update on StaffInfo to groomer_group;
grant select on StaffContracts to groomer_group;
grant select on StaffSalaryChanges to groomer_group;
grant select on Pets_Groomer to groomer_group;
--Функції--
grant execute on function ServiceResourses_Groomer to
groomer_group;
--Завскладом--
--Створення групи--
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM
warehm_group;
drop role warehm_group;
create role warehm_group NOLOGIN;
--Таблиці та представлення--
grant select on StaffInfo to warehm_group;
grant update on StaffInfo to warehm_group;
grant select on StaffContracts to warehm_group;
grant select on StaffSalaryChanges to warehm_group;
grant select on Resourse to warehm_group;
grant select on ResStorage_WareHM to warehm_group;
--Функції--
grant execute on function ServiceResourses_WareHM to
warehm_group;
grant execute on function CostReport_WareHM to warehm_group;
grant execute on function ProcReport_WareHM to warehm_group;
--Менеджер--
--Створення групи--
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM
manag_group;
drop role manag_group;
create role manag_group NOLOGIN;
--Таблиці та представлення--
grant select on AllServicePets_Manag to manag_group;
grant select on AllPetTypesInfo_Manag to manag_group;
grant select, update on Service to manag_group;
grant select, update on ServicePet to manag_group;
grant select on Resourse to manag_group;
grant select on AllServPetRes_Manag to manag_group;
grant select, update, delete on Resourse to manag_group;
grant select, update on PetKind to manag_group;
grant select on AllRooms_Manag to manag_group;
grant select, update, delete on PetRoom to manag_group;
grant select on AllStaffInfo_Manag to manag_group;
```

Лістинг И.1, аркуш 2

```
grant select on AllStaffContracts_Manag to manag_group;
grant select on StaffSalaryChanges_Manag to manag_group;
grant select, update, delete on SalaryChange to manag_group;
grant select, update, delete on StaffContract to manag_group;
grant select, update on Position to manag_group;
grant select on Analitic_ClientRating to manag_group;
grant select on Analitic_ProcReports to manag_group;
grant select on Analitic_ClientComments to manag_group;
grant select on AllowedPetType to manag_group;
--Функції--
grant execute on function AddEditTypeServ_Manag to manag_group;
grant execute on function AddNewService_Manag to manag_group;
grant execute on function FindAvgResAmount_Manag to manag_group;
grant execute on function AddEditServPetRes_Manag to
manag_group;
grant execute on function DeleteServPetRes_Manag to manag_group;
grant execute on function EditPetType to manag_group;
grant execute on function AddNewPetType to manag_group;
grant execute on function AddNewRoom to manag_group;
grant execute on function AddSalChange_Manag to manag_group;
grant execute on function GiveRole_Manag to manag_group;
grant execute on function TakeRole_Manag to manag_group;
grant execute on function StaffReg_Manag to manag_group;
grant execute on function Analitic_MonthProfit to manag_group;
grant execute on function Analitic_PopularServices to
manag_group;
grant execute on function Analitic_ServCostReport to
manag_group;
grant execute on function Analitic_StaffReserv to manag_group;
--Ветеринар--
--Створення групи--
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM
vet_group; drop role vet_group;
create role vet_group NOLOGIN;
--Таблиці та представлення--
grant select, update on StaffInfo to vet_group;
grant select on StaffContracts to vet_group;
grant select on StaffSalaryChanges to vet_group;
grant select on AllReservs_Vet to vet_group;
grant select, update, delete on AnalysisType to vet_group;
grant select, update, delete on AnalysisPetType to vet_group;
grant select on AllPetKind_Vet to vet_group;
grant select on AllAnalysisPetType_Vet to vet_group;
grant select on Analysis to vet_group;
grant select on AllMedResourses_Vet to vet_group;
--Функції--
grant execute on function AssignRreserv_Vet to vet_group;
grant execute on function EditMedPrescription_Vet to vet_group;
grant execute on function AddAnalysisType_Vet to vet_group;
grant execute on function AddAnalysisPetType_Vet to vet_group;
grant execute on function AddAnalysis_Vet to vet_group;
```