МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії
програмного забезпечення

_____ Євген ДАВИДЕНКО

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК ВЕТЕРИНАРНОЇ КЛІНІКИ

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Інженерія програмного забезпечення»

Здобувачка _____ **Марина ШИШКАНОВА**

«__» _____ 20__ р.

Керівник роботи
PhD, старший
викладач _____ **Ігор КАНДИБА**

«__» _____ 20__ р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«____» _____ 2026 р.

ЗАВДАННЯ
на кваліфікаційну бакалаврську роботу здобувача
Шишканова Марина

1. Тема кваліфікаційної роботи Вебзастосунок ветеринарної клініки затверджена наказом ректора ЧНУ ім. Петра Могили № _349__ від «_26_» _грудня_ 2026 р.
2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.
3. Результатом роботи є створений вебзастосунок ветеринарної клініки, який може бути використаний у ветеринарних клініках для організації роботи приймальної, надання онлайн-консультацій, ведення обліку пацієнтів та підвищення рівня обслуговування клієнтів.

4. Перелік питань, що підлягають розробці:

— Проаналізувати існуючі вебресурси ветеринарних клінік для виявлення їх переваг та недоліків.

— Визначити функціональні вимоги до сайту та необхідні модулі.

— Розробити структуру сайту.

— Створити прототип інтерфейсу користувача.

— Реалізувати основний функціонал сайту.

5. Перелік графічних матеріалів: Презентація

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «____» _____ 20___ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: **Вебзастосунок ветиринарної клініки**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	08.02.26	14.02.26	Виконано
2.	Огляд літератури за темою роботи	14.02.26	20.02.26	Виконано
3.	Складання календарного плану КБР	20.02.26	23.02.26	Виконано
4.	Аналіз предметної області	23.02.26	25.02.26	Виконано
5.	Розробка проєктних рішень	25.03.26	05.04.26	Виконано
6.	Моделювання та конструювання ПЗ	05.04.26	01.05.26	Виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	01.05.26	25.06.26	Виконано
8.	Відгук керівника КБР	20.04.26	03.05.26	Виконано
9.	Оформлення КБР та презентації	03.05.26	25.05.26	Виконано
10.	Попередній захист	25.05.26	26.05.26	Виконано
11.	Рецензування	12.06.26	18.06.26	Виконано
12.	Завершення оформлення КБР та презентації	26.05.26	12.06.26	Виконано
13.	Захист кваліфікаційної роботи			

Здобувачка _____ **Марина ШИШКАНОВА**

«__» _____ 20__ р.

Керівник роботи

PhD, старший викладач _____ **Ігор КАНДИБА**

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

Вебзастосунок ветеринарної клініки

Здобувачка 408 гр.: Шишканова Марина

Керівник: PhD, старший викладач Кандиба Ігор

Актуальність роботи полягає в тому, що у сучасних умовах розвитку інформаційних технологій вебсайти є одним з ключових інструментів комунікації та обслуговування клієнтів різних сферах діяльності та обслуговування, зокрема у ветеринарних клініках. Сайт ветеринарної клініки дозволяє не лише надавати інформацію про послуги та графік роботи, а й забезпечує онлайн-запис на прийом, консультації та облік пацієнтів.

Об’єктом кваліфікаційної роботи є процеси обслуговування клієнтів та організації роботи ветеринарної клініки.

Предметом кваліфікаційної роботи являється методи та засоби створення вебсайту для автоматизації запису та надання інформації про послуги ветеринарної клініки.

Метою роботи є розробка вебсайту ветеринарної клініки для обслуговування клієнтів, запису на прийом та комунікації клієнта з лікарем.

Кваліфікаційна робота складається із вступу, 3 розділів, висновків та переліку джерел посилання.

У вступі детально описано актуальність теми, визначено об’єкт та предмет дослідження, а також сформовано мету.

У першому розділі було проведено аналіз предметної області та існуючих аналогів вебсайтів ветеринарних клінік, розглянуто їх плюси та мінуси, сформовано подальші кроки розробки вебзастосунку.

У другому розділі було проведено аналіз сучасного стану, обґрунтовано вибір технологій та специфікація вимог до вебзастосунку ветеринарної клініки

У третьому розділі розглянуто проєктуванню вебзастосунку, зокрема розробці структури сайту, ER-діаграми бази даних, UML-діаграм.

У четвертому розділі описано процес реалізації вебсайту, програмну частину, інтерфейс користувача, а також тестування розробленого застосунку.

У висновках підведено підсумки виконаної роботи, визначено досягнення поставленої мети та окреслено можливі напрями подальшого розвитку системи.

Кваліфікаційна робота викладена на 64 сторінках машинописного тексту, складається із вступу, 4 розділів, загальних висновків, переліку джерел посилання з 25 найменувань. Праця містить 3 таблиць та 39 рисунків.

Ключові слова: вебзастосунок, ветеринарна медицина, онлайн запис, облік пацієнтів, база даних, php.

ABSTRACT

to the qualifying bachelor's thesis

Web Application for a Veterinary Clinic

Student of 408 group: Shyshkanova Maryna

Supervisor: PhD, senior lecturer Kandyba Ihor

The relevance of the work lies in the fact that in modern conditions of information technology development, websites are one of the key tools for communication and customer service in various areas of activity and service, in particular in veterinary clinics. The veterinary clinic website allows not only to provide information about services and work schedules, but also provides online appointments, consultations and patient registration.

The object of the study is the processes of customer service and organization of the work of a veterinary clinic.

The subject of the study is methods and means of creating a website to automate booking and providing information about veterinary clinic services.

The purpose of the work is to develop a veterinary clinic website for customer service, appointment booking, and client-doctor communication.

The qualification work consists of an introduction, 3 sections, conclusions and a list of references.

The introduction describes in detail the relevance of the topic, defines the object and subject of the study, and formulates the goal.

The first chapter analyzed the subject area and existing similar websites for veterinary clinics, examined their pros and cons, and outlined the next steps in developing the web application.

The second chapter analyzed the current state of the field, justified the choice of technologies, and specified the requirements for the veterinary clinic’s web application.

The third chapter examined the design of the web application, specifically the development of the site structure, the database ER diagram, and the UML diagrams.

The fourth chapter describes the process of implementing the website, the software component, the user interface, and the testing of the developed application.

The conclusions summarize the work done, determine the achievement of the set goal, and outline possible directions for further development of the system.

The qualification work is presented on 64 pages of typewritten text, consists of an introduction, 4 sections, general conclusions, a list of references with __ titles. The work contains 25 tables and 3 figures.

Keywords: web application, veterinary medicine, online recording, patient registration, database.

ЗМІСТ

ВСТУП

В сучасних умовах розвитку інформаційних технологій вебсайти є важливим інструментом комунікації та обслуговування клієнтів у різних сферах діяльності, зокрема у ветеринарних клініках. Сайт ветеринарної клініки дозволяє не лише надавати інформацію про послуги та графік роботи, а й забезпечує онлайн-запис на прийом, консультації та облік пацієнтів.

Метою роботи є розробка вебсайту ветеринарної клініки для обслуговування клієнтів, запису на прийом на комунікації клієнта з лікарем.

Завдання:

1. Проаналізувати існуючі вебресурси ветеринарних клінік для виявлення їх переваг та недоліків.
2. Визначити функціональні вимоги до сайту та необхідні модулі (онлайн-запис, каталог послуг, контактна інформація).
3. Розробити структуру сайту та інформаційну модель (схему сторінок і зв’язків між ними).
4. Створити прототип інтерфейсу користувача.
5. Реалізувати основний функціонал сайту за допомогою сучасних вебтехнологій (HTML, CSS, JavaScript,Vue.js).
6. Провести тестування сайту на відповідність вимогам та зручність використання для клієнтів і персоналу клініки.

Об’єктом кваліфікаційної роботи є процеси обслуговування клієнтів та організації роботи ветеринарної клініки.

Предметом кваліфікаційної роботи являється методи та засоби створення вебсайту для автоматизації запису та надання інформації про послуги ветеринарної клініки.

Аналіз сучасного стану проблеми показує, що більшість ветеринарних клінік мають обмежені або застарілі вебресурси, які не забезпечують повного спектра послуг онлайн. Існує потреба у створенні сайту, який поєднує інформаційні, комунікаційні та управлінські функції.

Розробка сайту передбачає використання сучасних вебтехнологій (HTML, CSS, JavaScript) та фреймворків для забезпечення адаптивності та інтерактивності. Впровадження системи онлайн-запису та керування послугами дозволить покращити взаємодію клініки з клієнтами.

Результати розробки можуть бути використані у ветеринарних клініках для організації роботи приймальної, надання онлайн-консультацій, ведення обліку пацієнтів та підвищення рівня обслуговування клієнтів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ АНАЛОГІВ ВЕБСАЙТІВ ВЕТЕРИНАРНИХ КЛІНІК

1.1 Актуальність та науково-практичне значення теми

На сьогоднішній день ветеринарна клініка – це багатопрофільний заклад, що надає широкий спектр послуг: профілактичні огляди, діагностику, хірургічні втручання, вакцинацію, лабораторні дослідження, стаціонарне лікування та реабілітацію тварин. Щоденна робота клініки супроводжується значним обсягом операцій: запис клієнтів на прийом, ведення медичних карток, планування графіків лікарів, облік фінансових операцій, формування звітності. Водночас, як зазначають дослідники [6, 20], більшість ветеринарних закладів в Україні досі використовують паперові журнали, розрізнені Excel-файли або застарілі локальні програми, що не забезпечують ні належної швидкості обслуговування, ні цілісності даних, ні зручного доступу для клієнтів.

Сучасний стан розвитку інформаційних технологій кардинально змінив очікування споживачів медичних послуг, зокрема ветеринарних. Як показує аналіз, проведений у роботах [9, 10], власники домашніх тварин дедалі частіше надають перевагу клінікам, які пропонують онлайн-запис, електронні нагадування про візити, можливість перегляду історії лікування через особистий кабінет та дистанційні консультації. Відсутність таких сервісів не лише знижує лояльність клієнтів, але й призводить до втрати конкурентних переваг [15]. Відтак, вебсайт ветеринарної клініки вже давно не виконує роль простої «візитівки». Зараз це має бути повноцінний інструмент управління взаємовідносинами з клієнтами (CRM) і надання сервісів онлайн [13].

Основними потребами, які висуваються до сучасного вебзастосунку для ветеринарної клініки, є:

- автоматичне нагадування клієнтам про майбутні візити та планові вакцинації;
- швидкий та зручний онлайн-запис на прийом з інтерактивним календарем;

- централізоване зберігання медичних карток тварин та історій лікування;
- можливість онлайн-оплати послуг та формування електронних чеків;
- управління персоналом із розмежуванням прав доступу (адміністратор, лікар, клієнт);
- генерація статистичних звітів для адміністрації (завантаженість лікарів, виручка, кількість прийомів).

Актуальність розроблення власного вебзастосунку підтверджується також світовими тенденціями. Згідно з аналітичними звітами [4, 13], ринок програмного забезпечення для ветеринарної медицини зростає в середньому на 8–12 % щорічно. Впровадження таких систем дозволяє скоротити час обслуговування одного клієнта на 30–40 %, зменшити кількість пропущених візитів через автоматичні нагадування та підвищити загальну задоволеність клієнтів [1, 17].

Науково-практичне значення теми полягає в тому, що в рамках роботи буде створено не просто статичний сайт, а функціональну клієнт-серверну систему, яка поєднує:

- реляційну базу даних (MS SQL Server) для надійного зберігання медичної та фінансової інформації;
- бекенд на PHP, що реалізує бізнес-логіку (запис, редагування карток, формування рахунків);
- фронтенд на Vue.js з адаптивним дизайном, який забезпечує комфортну роботу як з ПК, так і з мобільних пристроїв.

Отримані результати можуть бути використані як типовий шаблон для автоматизації невеликих та середніх ветеринарних клінік, а також як навчальний приклад при вивченні дисциплін «Web-технології», «Проєктування баз даних» та «Розробка програмного забезпечення». Крім того, запропонована архітектура дозволяє легко масштабувати систему впроваджувати перспективні напрямки, такі як телеветеринарія [10, 13].

1.2 Аналіз існуючих програмних рішень для ветеринарних клінік

Перед розробленням власної системи потрібно дослідити функціональні можливості та обмеження існуючих вебсайтів ветеринарних клінік. Нижче наведено аналіз двох характерних представників: простого іміджевого сайту та багатофункціональної платформи.

Таблиця 1.1 – Програмний аналог velesvetclinic [21]

Архітектура	Web application
Мови реалізації	HTML, CSS, JavaScript
Перелік функцій	1. Перелік послуги, 2. контактна інформація
Переваги	Простий інтерфейс
Недоліки	Обмежений функціонал

Даний сайт виконує роль візитівки, відвідувач може ознайомитися з переліком послуг та контактною інформацією. Будь-які інтерактивні функції відсутні, що робить його непридатним для автоматизації роботи клініки.

Таблиця 1.2 – Програмний аналог ЕКСВЕТ [22]

Архітектура	Client-server (вебклієнт + сервер)
Мови реалізації	PHP, JavaScript, HTML/CSS; база даних MySQL
Перелік функцій	1. Перелік послуг 2. Форма для звернення 3. Інформація про лікарів 4. Перегляд новин і статей
Переваги	Зручний інтерфейс для клієнтів
Недоліки	1. Відсутність особистих кабінетів користувачів Немає системи онлайн-запису

ЕКСВЕТ має ширший спектр можливостей, форму для звернення, перегляд новин і статей. Проте ця система має суттєві обмеження – відуктність онлай запису, введення карток пацієнтів. Це створює передумови для створення програмного забезпечення, яке усуне недоліки та розширить перелік доступних сервісів.

На основі вивчення предметної області та аналізу аналогів сформульовано розширений перелік функцій, які мають бути реалізовані у власній розробці:

1. Реєстрація та авторизація користувачів.
2. Розмежування ролей (адміністратор, лікар, клієнт).
3. Створення та ведення медичних карток тварин.
4. Історія прийомів та лікування.
5. Планування та календар прийомів.
6. Онлайн-запис на прийом.
7. Нагадування клієнтам про прийоми.
8. Оплата послуг онлайн.
9. Формування рахунків та чеків.
10. Каталог послуг клініки.
11. Пошук лікаря за спеціалізацією.
12. Генерація статистичних звітів.
13. Управління працівниками клініки.
14. Інтеграція з Email.
15. Ведення історії вакцинацій.
16. Управління розкладом роботи лікарів.
17. Формування рекомендацій лікаря.
18. Можливість завантаження документів та результатів аналізів.
19. Перегляд інформації про лікарів.
20. Нагадування лікарям про нові записи.

1.3 Обґрунтування необхідності розробки

Проведений у підрозділах 1.1–1.2 аналіз предметної області та існуючих програмних рішень для ветеринарних клінік дозволяє зробити висновок про

наявність суттєвого розриву між потребами сучасних ветеринарних закладів та функціональними можливостями типових сайтів-візиток або застарілих систем. Як зазначають дослідники [6; 20], більшість доступних рішень або обмежуються пасивним інформуванням, або є дорогими коробковими продуктами, не адаптованими до українського ринку та специфіки роботи невеликих клінік.

Виявлені недоліки аналогів, зокрема відсутність:

- онлайн-запису на прийом з інтерактивним календарем [1; 17];
- ведення електронних медичних карток тварин та історії хвороб [9];
- автоматичних нагадувань клієнтам про візити та вакцинації [4];
- розмежування ролей (клієнт, лікар, адміністратор) з відповідними правами доступу [12];
- можливості онлайн-оплати послуг та генерації фінансових звітів [3];

обумовлюють необхідність створення нового вебзастосунку, який би комплексно вирішував зазначені проблеми.

Відповідно до мети кваліфікаційної роботи – розробка вебзастосунку ветеринарної клініки для обслуговування клієнтів, запису на прийом та комунікації клієнта з лікарем – сформульовано наступні завдання, які необхідно вирішити в процесі виконання роботи:

1. Розробити архітектуру вебзастосунку на основі клієнт-серверної моделі з використанням REST API, що забезпечить модульність, масштабованість та можливість подальшої інтеграції з мобільними додатками [12; 18].

2. Спроектувати реляційну базу даних в середовищі MS SQL Server, яка б забезпечувала цілісне зберігання інформації про: користувачів (з ролями), тварин (з прив’язкою до власників), лікарів (спеціалізація, стаж), послуги (ціни, тривалість), записи на прийом, медичні картки, платежі та сповіщення [5; 18]. Передбачити механізми захисту від SQL-ін’єкцій та несанкціонованого доступу.

3. Реалізувати серверну логіку мовою PHP з використанням об’єктно-орієнтованого підходу та PDO для взаємодії з базою даних. Забезпечити:

- реєстрацію, автентифікацію та керування сесіями користувачів;

– створення, редагування, видалення записів на прийом (з перевіркою доступності часу);

– ведення медичних карток (діагнози, лікування, завантаження файлів результатів аналізів) [9];

– автоматичне надсилання електронних листів (PHPMailer) про підтвердження запису, нагадування, зміну статусу [14].

4. Розробити клієнтську частину (фронтенд) на основі фреймворку Vue.js з використанням Vue Router та Axios [11; 19]. Забезпечити:

– інтерактивний календар для вибору вільного часу запису;

– особистий кабінет клієнта (перелік тварин, історія прийомів, медичні картки, оплата);

– робоче місце лікаря (розклад прийомів, доступ до карток пацієнтів, призначення лікування);

– адміністративну панель (керування користувачами, послугами, перегляд звітів).

5. Провести тестування розробленого програмного забезпечення

6. Скласти керівництво користувача з описом основних сценаріїв роботи для кожної ролі (клієнт, лікар, адміністратор).

Виконання перелічених завдань дозволить створити повнофункціональний вебзастосунок, який усуває недоліки існуючих аналогів, забезпечує автоматизацію основних процесів ветеринарної клініки та підвищує якість обслуговування клієнтів. Отримані результати стануть основою для подальшого проєктування, реалізації та тестування системи у наступних розділах кваліфікаційної роботи.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи проведено системний аналіз предметної області ветеринарних клінік, досліджено сучасний стан автоматизації їхньої діяльності та обґрунтовано необхідність створення власного вебзастосунку.

У підрозділі 1.1 розкрито актуальність теми, яка полягає у зростаючій потребі ветеринарних закладів у цифровізації основних процесів: онлайн-запису на прийом,

ведення електронних медичних карток, автоматичних нагадувань, онлайн-оплати та генерації звітів. На основі аналізу літературних джерел визначено ключові вимоги до сучасного вебзастосунку та доведено його науково-практичне значення.

У підрозділі 1.2 проведено огляд та порівняльний аналіз двох існуючих програмних рішень: сайту-візитки «Veles Vet Clinic» (обмежений функціонал, відсутність інтерактивності) та системи «ЕКСВЕТ» (ширші можливості, але немає онлайн-запису, особистих кабінетів і медичних карток). Виявлені недоліки підтвердили доцільність створення нового вебзастосунку, який би комплексно автоматизував роботу клініки.

У підрозділі 1.3 на основі аналізу предметної області та виявлених прогалин сформульовано конкретні завдання кваліфікаційної роботи: розробка архітектури клієнт-серверного застосунку, проєктування бази даних MS SQL Server, реалізація бекенду на PHP (автентифікація, записи, картки, оплата, повідомлення), створення фронтенду на Vue.js (адаптивний інтерфейс, календар, особисті кабінети, адмінпанель), тестування та складання керівництва користувача.

Таким чином, результати першого розділу створюють повне уявлення про об’єкт автоматизації, вимоги до системи та шляхи їх реалізації, що є достатньою основою для переходу до наступних етапів роботи – вибору технологій, проєктування та безпосередньої розробки вебзастосунку ветеринарної клініки.

2 АНАЛІЗ СУЧАСНОГО СТАНУ, ОБГРУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ТА СПЕЦИФІКАЦІЯ ВИМОГ ДО ВЕБЗАСТОСУНКУ ВЕТЕРИНАРНОЇ КЛІНІКИ

2.1 Сучасні тенденції розроблення вебзастосунків для ветеринарного бізнесу

Сьогодні ринок програмного забезпечення для ветеринарних клінік перебуває на стадії активного зростання [22]. Все більше закладів залишають осторонь паперові документи та розрізнені Excel-файли, обираючи на користь сучасних централізованих і веборієнтованих систем. Однак більшість доступних сьогодні рішень стикаються з низкою суттєвих обмежень. Одні пропонують занадто вузький функціонал, наприклад, лише простий каталог із контактною інформацією, тоді як інші є дорогими коробковими продуктами з мінімальними можливостями для налаштування під конкретні потреби [1]. Серед головних недоліків таких систем варто виділити відсутнрість онлайн запису, оплати онлайн, управління персоналом, нагадування про запис, керування медичними картками. Додатково, нерідко зустрічаються заплутаний інтерфейс і недостатня зручність для користувачів.

Водночас очікування власників домашніх тварин продовжують зростати. Вони все більше цінують такі можливості, як онлайн-запис на прийом [9], автоматичні нагадування про візити чи процедури та зручні способи безготівкової оплати. Це створює чіткий попит на універсальний, гнучкий і масштабований вебзастосунок, який поєднає управління внутрішніми процесами клініки, обслуговування клієнтів і системи аналітики в рамках єдиної інтегрованої платформи [22].

У першому розділі було визначено перелік ключових функцій, які повинні скласти основу даного рішення. Для того щоб забезпечити успішну реалізацію такої системи, необхідно обрати сучасний і продуктивний технологічний стек, який сприятиме високій надійності, безпеці та полегшить підтримку програмного продукту.

2.1.1 Серверна частина: PHP та MS SQL Server

PHP є однією з найпоширеніших мов для серверної веброзробки. Це підтверджується великою кількістю готових бібліотек, фреймворків і хостингів, які її підтримують [10]. Для ветеринарної клініки, де не потрібні складні обчислення в реальному часі, PHP забезпечує достатню швидкодію та дозволяє швидко реалізувати бізнес-логіку у вигляді REST-контролерів [1]. Важливо, що PHP має гарну підтримку роботи з MS SQL Server через драйвери PDO та sqlsrv, що дозволяє використовувати параметризовані запити та убезпечити систему від SQL-ін'єкцій.

Вибір MS SQL Server як системи керування базами даних зумовлений кількома причинами. Це реляційна СКБД, яка гарантує цілісність даних завдяки транзакціям, зовнішнім ключам та обмеженням. Для медичних карток тварин, фінансових операцій та історії лікування така надійність є вкрай важливою. До того ж MS SQL Server має вбудовані засоби для створення звітів (SQL Server Reporting Services) та аналітики, що полегшує реалізацію статистичних звітів для адміністрації. Можливість використання збережених процедур і функцій дозволяє перенести частину бізнес-логіки на рівень бази даних, зменшуючи навантаження на сервер застосунку. Крім того, гнучке налаштування безпеки на рівні користувачів та ролей СКБД додає додатковий захист конфіденційної інформації.

2.1.2 Технології розробки клієнтської частини вебзастосунку

Для створення динамічного та чуйного інтерфейсу обрано стек на основі JavaScript. Ця мова є стандартом для браузерного середовища.

Основним фронтенд-фреймворком виступає Vue.js. Він поєднує простоту вивчення з потужною реактивною системою, компонентним підходом та віртуальним DOM, що забезпечує високу швидкість оновлення інтерфейсу. Завдяки однофайловим компонентам шаблон, логіка та стилі кожного елемента зберігаються в одному місці, що зручно при розробці та підтримці коду. Vue Router дає змогу організувати навігацію всередині односторінкового застосунку без перезавантаження сторінки, створюючи враження роботи з нативною програмою.

Це особливо корисно для сценаріїв, коли реєстратор або лікар часто переміщується між розділами – календарем, картками пацієнтів, рахунками.

Для обміну даними з серверним API використовується бібліотека Axios. Вона працює на основі промісів, підтримує автоматичне перетворення JSON, налаштування перехоплювачів для додавання токенів авторизації до кожного запиту та централізовану обробку помилок. Це значно спрощує реалізацію безпечної взаємодії клієнта з бекендом, а також дозволяє уніфікувати логіку повторних спроб у разі збоїв мережі.

Поєднання Vue.js і Axios дає легкий, але функціональний фронтенд, який швидко завантажується, мінімально навантажує сервер і легко адаптується для мобільних пристроїв за допомогою адаптивної верстки.

2.2 Специфікація вимог до вебзастосунку

Моделювання предметної області є важливим етапом розробки програмного забезпечення, оскільки дозволяє визначити структуру системи та встановити взаємозв’язки між її компонентами. У рамках даної роботи моделювання здійснюється для процесів управління клієнтами, тваринами, лікарями, записами на прийом та медичними картками у вебзастосунку ветеринарної клініки.

Основними акторами системи є клієнт (власник тварини), лікар, адміністратор, реєстратор та гість. Клієнт здійснює онлайн-запис на прийом, переглядає історію візитів своїх тварин, редагує інформацію про тварин та оплачує рахунки. Лікар веде медичні картки, призначає лікування та переглядає історію лікування тварини. Адміністратор керує персоналом, довідниками послуг, генерує фінансові та статистичні звіти, а також виконує аудит дій користувачів. Гість має змогу переглядати каталог послуг та інформацію про лікарів без авторизації.

У межах моделювання визначаються основні сутності системи, серед яких користувачі (з ролями), тварини, медичні картки, лікарі, послуги, записи на прийом, оплата, сповіщення. Для кожної сутності визначаються атрибути та зв’язки, що дозволяє сформувати логічну модель бази даних.

Важливим етапом є формування специфікації вимог до програмного забезпечення (SRS), яка визначає функціональні та нефункціональні вимоги до системи. Специфікація вимог є основою для подальшої розробки, тестування та впровадження програмного продукту.

1) Призначення та межі проєкту:

1.1) призначення системи: вебзастосунок ветеринарної клініки призначений для автоматизації процесів запису клієнтів на прийом, ведення медичних карток тварин, обліку наданих послуг, управління персоналом та формування фінансової звітності. Система забезпечує взаємодію між клієнтами (власниками тварин), лікарями та адміністраторами.

1.2) Погодження, що ухвалені в програмній документації:

− система реалізується як вебзастосунок;

− технології: HTML, CSS, JavaScript (Vue.js), PHP 8.1;

− база даних: MS SQL Server 2022.

1.3) Межі проєкту ПЗ:

− розробка охоплює вебінтерфейс та серверну логіку;

− включає модулі: автентифікація, користувачі та ролі, клієнти, тварини, медичні картки, лікарі, послуги, записи на прийом, рахунки, сповіщення;

− не передбачено мобільний застосунок (але інтерфейс адаптивний).

2) Загальний опис:

2.1) сфера застосування: система застосовується у ветеринарних клініках для автоматизації запису пацієнтів, ведення електронних медичних карток тварин, управління розкладом лікарів та фінансового обліку.

2.2) Характеристики користувачів:

− Адміністратор – керування системою: управління персоналом, послугами, генерація звітів, перегляд усіх записів і карток;

− Лікар – перегляд розкладу, ведення медичних карток;

− Клієнт – онлайн-запис на прийом, додавання та редагування тварин, онлайн-оплата;

− Гість –перегляд каталогу послуг та інформації про лікарів без реєстрації.

2.3) Загальна структура і склад системи:

− клієнтська частина (HTML, CSS, Vue.js);

− серверна частина (PHP 8.1);

− база даних (MS SQL Server).

2.4) Загальні обмеження:

− система функціонує лише за стабільного інтернет-з’єднання;

− не реалізовано підтримку справжніх платежів або інтеграцію з банківськими системами;

− одночасна кількість користувачів не менше 50.

3) Функції системи

3.1) Управління користувачами та автентифікація:

Опис функції: забезпечує реєстрацію, вхід у систему, призначення ролей (адміністратор, лікар, клієнт), а також розмежування доступу.

Вхідна інформація: логін, пароль, роль, email, контактні дані.

Вихідна інформація: запис у БД, статус автентифікації, токен.

Функціональні вимоги: перевірка унікальності логіну, хешування паролів (bcrypt), підтримка сесій, захист від CSRF, XSS.

3.2) Управління клієнтами та тваринами:

Опис функції: ведення бази клієнтів та їхніх тварин. Клієнт самостійно додає,редагує та видаляє своїх тварин.

Вхідна інформація: Ім’я користувача, телефон, email; для тварини – кличка, вид, вік, особливі позначки.

Вихідна інформація: запис у БД, список тварин у кабінеті клієнта.

Функціональні вимоги: пошук клієнтів (для адміністратора), валідація введених даних, прив'язка тварин до клієнта.

3.3) Управління записами на прийом:

Опис функції: онлайн-запис клієнта до лікаря з вибором послуги, дати та часу через інтерактивний календар. Скасування запису – клієнтом або адміністратором.

Вхідна інформація: клієнт, обраний лікар, послуга, тварина, дата, час.

Вихідна інформація: запис у календарі, підтвердження.

Функціональні вимоги: перевірка зайнятості лікаря, автоматичне надсилання нагадувань, можливість скасування запису.

3.4) Ведення медичних карток тварин:

Опис функції: лікар створює та редагує медичну картку: діагнози, призначення, історію хвороб, результати аналізів.

Вхідна інформація: діагноз, призначення, дата прийому, нотатки.

Вихідна інформація: запис у БД, історія хвороби, доступна для перегляду клієнту.

Функціональні вимоги: розмежування доступу.

3.5) Управління послугами:

Опис функції: адміністратор додає, редагує та видаляє послуги (назва, ціна, тривалість, лікарі, що робить процедуру).

Вхідна інформація: назва послуги, ціна, тривалість, лікарі, що робить процедуру.

Вихідна інформація: послуги доступні для перегляду всім ролям.

Функціональні вимоги: фільтрація лікарів за спеціалізацією, валідація цін.

3.6) Звітність та аналітика:

Опис функції: адміністратор генерує фінансові звіти (виручка, кількість прийомів за період) та статистичні звіти (завантаженість лікарів, кількість пацієнтів).

Вхідна інформація: період, тип звіту.

Вихідна інформація: звіт за обраний період.

Функціональні вимоги: фільтрація за датою, лікарем, послугою.

3.7) Сповіщення:

Опис функції: автоматичне надсилання клієнтам підтвердження запису, нагадувань про прийом (email), а також сповіщень про скасування.

Вхідна інформація: email клієнта, тип події.

Вихідна інформація: відправлене повідомлення.

Функціональні вимоги: налаштовувані шаблони, журнал відправлень.

4) Вимоги до інформаційного забезпечення:

4.1) Джерела і зміст вхідної інформації (даних):

- дані користувачів системи (логін, хеш пароля, роль, контакти);
- інформація про клієнтів (власників тварин);
- дані про тварин (кличка, вид, вік);
- медичні картки (діагнози, призначення, історія);
- дані про лікарів (спеціалізація, стаж, графік);
- довідник послуг (назва, ціна, тривалість);
- записи на прийом (клієнт, лікар, тварина, дата, час, статус);
- рахунки та платежі.

4.2) Нормативно-довідкова інформація:

- довідник ролей користувачів;
- довідник статусів записів (заплановано, виконано, скасовано);

4.3) Вимоги до способів організації, збереження та ведення інформації

- зберігання даних у MS SQL Server;
- використання реляційної структури БД з нормалізацією;
- дані про тварин, профілів користувачів вводяться вручну.

5) Вимоги до технічного забезпечення:

- підтримка HTTPS;
- клієнтська частина: підтримка останніх браузерів.

6) Вимоги до програмного забезпечення:

6.1) Архітектура програмної системи: клієнт-серверна з використанням сесій, REST API (PHP).

6.2) Системне програмне забезпечення: Windows Server 2019+ або сумісне середовище

6.3) Мережне програмне забезпечення: HTTPS (TLS 1.2+).

6.4) Програмне забезпечення ведення інформаційної бази: MS SQL Server 2022.

6.5) Мова і технологія розробки ПЗ:

- бекенд: PHP 8.1;
- фронтенд: HTML5, CSS3, JavaScript.

7) Вимоги до зовнішніх інтерфейсів:

7.1) Інтерфейс користувача: вебінтерфейс з адаптивним дизайном (десктоп, планшет, смартфон).

7.2) Апаратний інтерфейс: стандартний ПК або ноутбук (клієнт), планшет, смартфон.

7.3) Програмний інтерфейс: REST API (PHP-скрипти, що повертають JSON), параметризовані SQL-запити (PDO).

7.4) Комунікаційний протокол: HTTPS.

8) Властивості програмного забезпечення:

8.1) Доступність: система доступна через браузер 24/7 (окрім планових технічних робіт).

8.2) Супроводжуваність: модульна структура коду.

8.3) Переносимість: може працювати на різних ОС за умови наявності сумісного вебсервера з PHP 8.1 та MS SQL Server (або альтернативної системи керування базами даних з мінімальними доробками).

8.4) Продуктивність: час відгуку на основні операції не більше 3 секунд, підтримка одночасної роботи не менше 50 користувачів.

8.5) Надійність: час відновлення після аварійного збою – до 5 хвилин. Регулярне резервне копіювання БД.

8.6) Безпека: HTTPS, хешування паролів bcrypt, захист від SQL-ін’єкцій (PDO), XSS, CSRF. сесії з безпечними налаштуваннями.

9) Інші вимоги:

- захист від SQL-ін’єкцій та інших вебатак;
- можливість розширення функціоналу (додавання нових типів звітів, інтеграція з лабораторними системами) без рефакторингу ядра;
- наявність інструкції користувача та документації адміністратора.

Висновки до розділу 2

У другому розділі проведено аналіз сучасного стану автоматизації ветеринарних клінік. Виявлено, що між зростаючими потребами закладів у

цифровізації та можливостями наявних програмних продуктів існує значний розрив. Більшість типових рішень обмежені презентаційними функціями. На основі цього обґрунтовано вибір технологічного стеку: серверна частина на PHP з використанням MS SQL Server як надійної СКБД, клієнтська частина на JavaScript з фреймворком Vue.js та бібліотекою Axios. Доведено, що таке поєднання забезпечує баланс продуктивності, безпеки, зручності супроводу та можливості розширення.

Сформульовано повний перелік функціональних і нефункціональних вимог до вебзастосунку, який охоплює всі ключові аспекти роботи ветеринарної клініки: від онлайн-запису та ведення медичних карток до фінансового обліку, звітності та управління персоналом. Окрему увагу приділено вимогам безпеки, надійності, продуктивності та зручності використання. Отримані результати слугують теоретичною та технічною основою для подальшого практичного проєктування архітектури системи, моделювання бази даних та безпосередньої розробки вебзастосунку.

Крім того, варто зазначити, що наведені висновки повністю узгоджуються з сучасними напрямками розвитку вебтехнологій у медичній сфері. Обраний підхід дозволяє не лише створити працездатний продукт у стислі терміни, але й закладає фундамент для подальшої модернізації, наприклад, впровадження мобільних додатків для клієнтів. Таким чином, результати аналізу та обґрунтування, отримані в цьому розділі, можна вважати достатніми для переходу до наступних етапів кваліфікаційної роботи.

3 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ВЕТЕРИНАРНОЇ КЛІНІКИ

3.1 Проєктування бази даних

Центральним елементом системи є реляційна база даних, створена в середовищі MS SQL Server. ER-діаграма (Entity-Relationship Diagram) створена для логічного моделювання бази даних вебзастосунку ветеринарної клініки і відображає основні сутності, їх атрибути та взаємозв'язки між ними (рисунок 3.1).

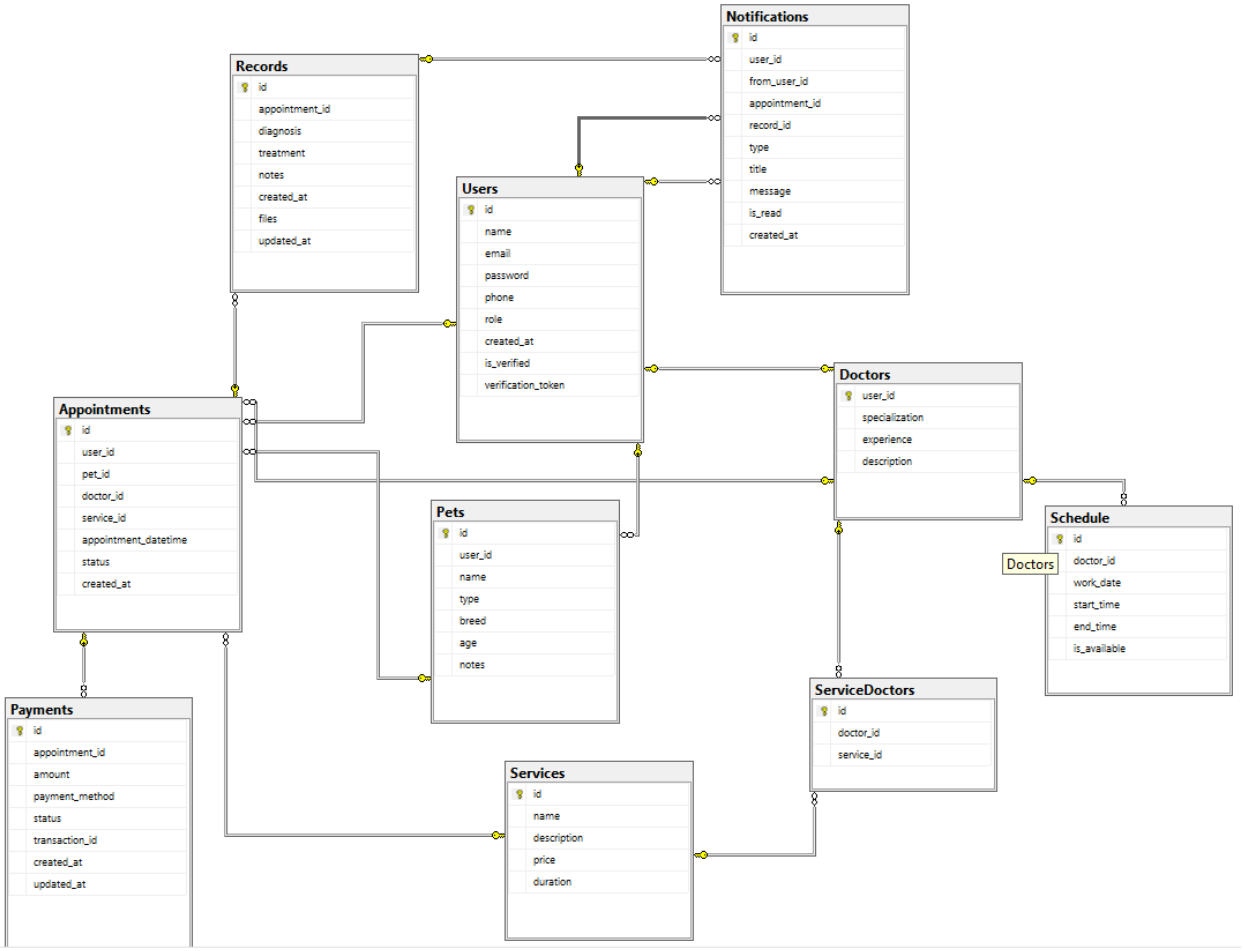

Рисунок 3.1 − ER-діаграма

Діаграма створена з урахуванням вимог до системи, зокрема необхідності зберігання інформації про користувачів, послуги, тварин, лікарів, записи та розклад роботи лікаря, а також забезпечення цілісності даних і ефективності виконання запитів.

Таблиця Users містить дані про всіх користувачів: адміністраторів, лікарів і клієнтів. Тут зберігаються відомості про ім'я, електронну пошту, пароль, телефон,

роль, дату створення облікового запису та параметри підтвердження особи. Один користувач може мати декілька тварин та записуватися на різні процедури. Зв’язки з таблицями Pets та Appointments реалізовано за типом «один до багатьох».

Таблиця Pets містить інформації про домашніх улюбленців користувачів. Кожен запис включає ім’я тварини, вид, породу, вік і додаткові дані. Між Users та Pets зв’язок «один до багатьох», оскільки одна тварина належить лише одному користувачу.

Таблиця Doctors зберігає дані про ветеринарів клініки. Для кожного лікаря вказані спеціалізація, стаж роботи та опис.

В таблиці Services зберігається перелік усіх ветеринарних послуг, таку як назва, опис, вартість, тривалість. Зв’язок між лікарями й послугами створено за допомогою проміжної таблиці ServiceDoctors, яка дозволяє реалізувати зв'язок «багато до багатьох», що дозволяє одному лікарю надавати різні послуги і одна послуга може бути виконана різними спеціалістами.

Таблиця Schedule відповідає за розклад роботи лікарів, фіксуючи дату, час початку і завершення робочих годин лікарів та їх доступность. Кожен запис пов’язаний із конкретним медиком.

Appointments зберігає інформацію про:

1) користувача, який записався;
2) тварину яку записали;
3) лікаря, що виконує цю процедуру;
4) інформацію про процедуру на яку записався користувач;
5) дату та час запису;
6) статус цього запису;
7) дата створення запису.

Ця таблиця має зв'язки з таблицями Users, Doctors, Records, Pets, Services та Payments для підтвердження запису після оплати.

Таблиця Records є медичною карткою тварини, в ній фіксуються діагнози, лікування, примітки, прикріплені файли, а також дати створення і оновлення запису. Кожен запис прив'язаний до певного прийому.

Payments потрібна для оплат послуг клініки. В ній міститься сума, спосіб оплати, статус транзакції і дату. Кожен платіж пов’язаний із відповідним записом на прийом.

Таблиця Notifications є системою сповіщень, зберігаючи повідомлення про записи на прийом та його скасування, вказуючи одержувача, відправника, тип повідомлення, текст і статус прочитання повідомлення.

Створена ER-діаграма відображає повну структуру інформаційної системи ветеринарної клініки, що забезпечує керування користувачами, їх тваринами, лікарями, послугами, записами на прийом, медичними картками, оплатами та системою повідомлень.

3.2 UML-моделювання системи

Розроблений вебзастосунок створено на основі клієнт-серверної архітектури. Клієнтська частина відповідає за інтерфейс користувача, серверна частина за роботою з базою даних та обробку запитів.

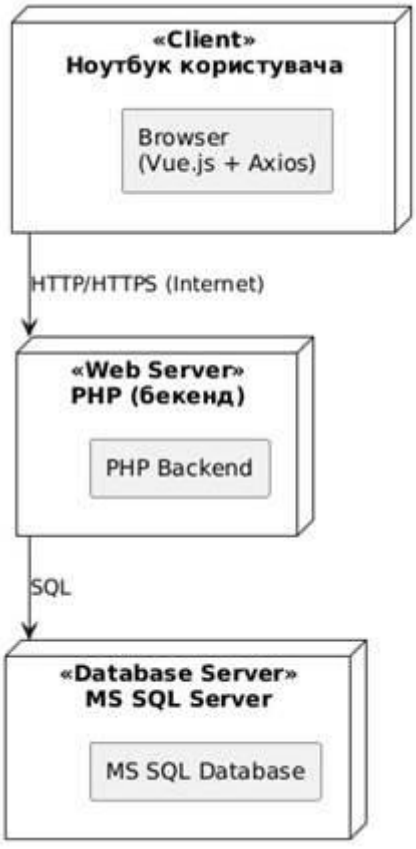

Рисунок 3.2 − Діаграма розгортання

На діаграмі (рис 3.2) зображено діаграму розгортання. Через браузер користувач робить HTTP-запити і надсилає їх до сервера. Сервер надсилає SQL запити до бази для створення та редагування картки тварин, історії хвороб, графіки роботи лікарів, ціни, даних клієнтів. Якщо бекенд потребує певні дані, він бере їх з SQL Server. База даних повертає записи, а бекенд вже віддає їх браузеру.

В результаті отримуємо три окремі частини:

1) браузер з фронтендом;
2) сервер з бекендом;
3) базу даних.

Ця схема називається трирівнева клієнт-серверною архітектура. Така система зручна тим, що кожну частину можна змінювати окремо, наприклад, оновити фронтенд без зупинки бази.

Діаграма прецедентів вебзастосунку ветеринарної клініки містить трьох акторів: адміністратор, лікар та звичайний користувач. Кожен тип користувача має свої особливі функції.

1) Адміністратор – керування розкладом персоналу, створює, редагує та видаляє послуги, доступ до всіх функцій системи.
2) Лікар – створює та редагує медичні картки пацієнтів, створює назначення пацієнтам.
3) Клієнт – записується та скасовує прийом, додає домашніх тварин, переглядає історії лікування, оплата послуг.

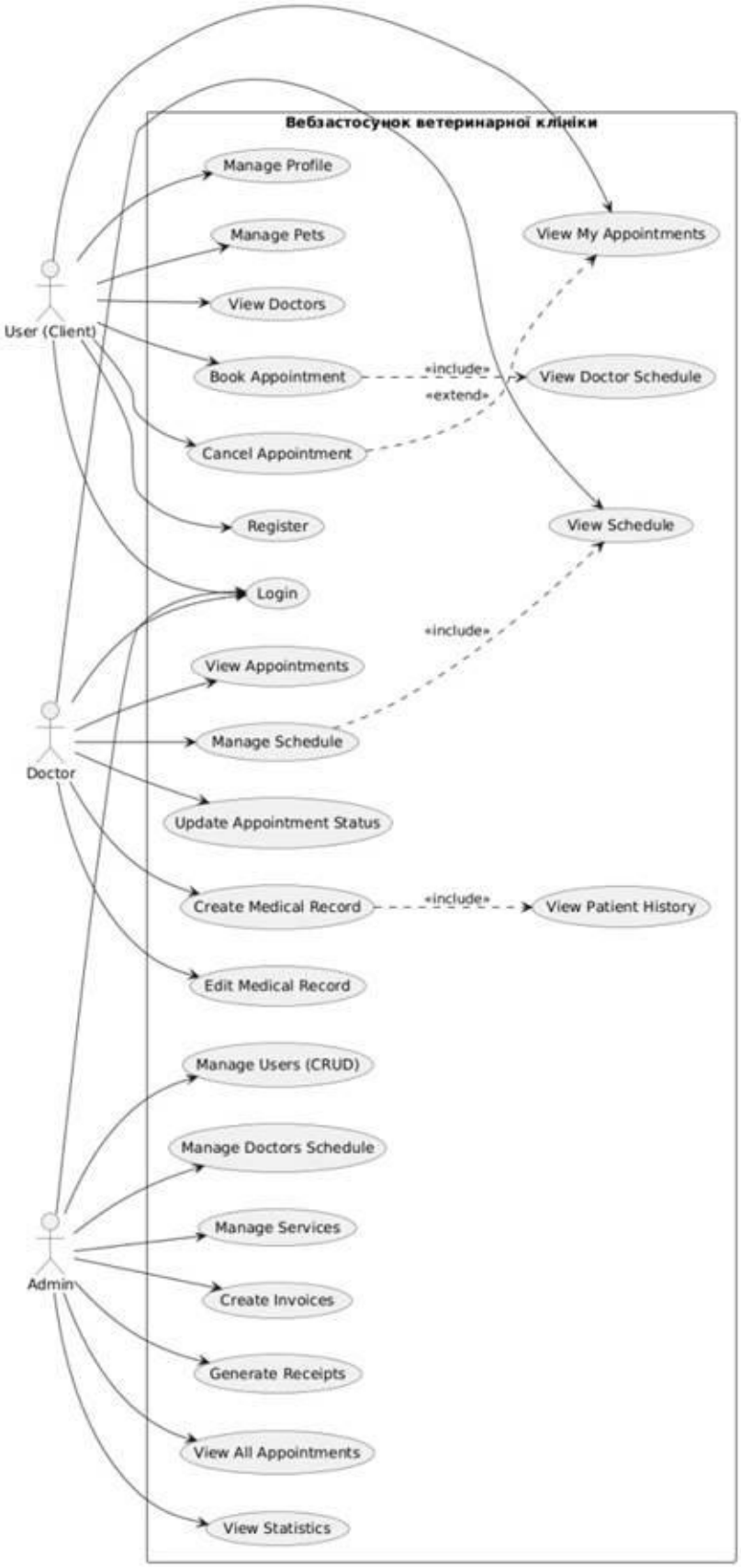

Рисунок 3.3– Діаграма прецедентів

Нижче подано детальний опис ключових варіантів використання (Use Cases), які охоплюють основні сценарії роботи із системою.

Use Case 1: Запис на прийом до ветеринара

Scope: Вебзастосунок для ветеринарної клініки

Level: User-goal

Primary Actor: Користувач (власник тварини)

Stakeholders and Interests:

1) Користувач: хоче записати тварину;
2) Ветеринар: зацікавлений у доступі до даних пацієнтів;
3) Адміністратор: контролює записи.

Preconditions:

1) Користувач зареєстрований та авторизований;
2) У системі є хоча б одна тварина користувача;
3) Вебзастосунок підключений до Інтернету.

Success Guarantee:

1) Запис на прийом створено;
2) Інформація збережена в базі даних;
3) Користувач отримав підтвердження запису.

Main Success Scenario:

1) Користувач входить у систему;
2) Система перевіряє дані та відкриває меню;
3) Користувач обирає «Запис на прийом»;
4) Користувач обирає тварину;
5) Користувач обирає ветеринара;
6) Система показує доступні дати та час;
7) Користувач обирає дату і час;
8) Система перевіряє доступність;
9) Система створює запис у базі;
10) Користувач отримує підтвердження.

Extensions (Alternative Scenarios):

1) Неправильний пароль – система пропонує повторити вхід;

2) Час зайнятий – система пропонує альтернативу;

3) Відсутній доступ до сервера – повідомлення «Не вдалося виконати операцію».

Special Requirements:

1) Передача даних через HTTPS;

2) Підтримка одночасної роботи кількох користувачів;

3) Час відгуку менше 3 секунд;

4) Автоматичні повідомлення (e-mail).

Use Case 2: Створення медичної картки та плану лікування

Scope: Вебзастосунок для ветеринарної клініки

Level: User-goal

Primary Actor: Лікар

Stakeholders and Interests:

1) Лікар: зацікавлений у веденні історії лікування;

2) Клієнт: отримує історію лікування;

3) Адміністратор: контроль ведення карток.

Preconditions:

1) Лікар авторизований;

2) Тварина зареєстрована у системі;

3) Тварина записана на процедуру;

4) Система доступна онлайн.

Success Guarantee:

1) Медична картка створена;

2) Історія лікування;

3) Клієнт отримав доступ до інформації.

Main Success Scenario:

1) Лікар входить у систему;

2) Система перевіряє права доступу;

3) Лікар обирає пацієнта;

4) Лікар створює нову медичну картку;
5) Лікар вносить історію лікування;
6) Лікар формує план вакцинацій;
7) Система зберігає дані в базі.

Extensions:

1) Пацієнт не зареєстрований – система пропонує створити картку;
2) Проблема з базою даних – повідомлення «Помилка збереження».

Special Requirements:

1) Захист даних пацієнтів (HTTPS, шифрування);
2) Підтримка одночасного редагування різними лікарями.

Use Case 3: Додавання нового користувача (лікаря)

Scope: Вебзастосунок для адміністрації клініки

Level: User-goal

Primary Actor: Адміністратор

Stakeholders and Interests:

1) Адміністратор: управляє персоналом;
2) Лікар: отримує доступ до системи.

Preconditions:

1) Адміністратор авторизований;
2) Наявність ролей у системі;
3) Система онлайн.

Success Guarantee:

1) Користувач доданий;
2) Призначені права доступу;
3) Користувач може увійти у систему.

Main Success Scenario:

1) Адміністратор входить у систему;
2) Обирає «Керування користувачами»;
3) Натискає «Додати користувача»;
4) Вводить дані користувача;

5) Призначає роль та права доступу;

6) Система зберігає дані;

7) Користувач отримує повідомлення про доступ.

Extensions:

1) Некоректні дані – система показує помилку;

2) Логін вже існує – система пропонує інший.

Use Case 4: Формування робочого графіку лікаря

Scope: Вебзастосунок ветеринарної клініки

Level: User-goal

Primary Actor: Адміністратор

Stakeholders and Interests:

1) Адміністратор: створює та редагує графік роботи лікарів;

2) Лікар: отримує свій розклад для прийому пацієнтів;

3) Клієнт: бачить доступність лікаря при записі.

Preconditions:

1) Адміністратор авторизований;

2) У системі існують лікарі;

3) База даних доступна.

Success Guarantee:

1) Графік лікаря створено або оновлено;

2) Дані збережені в базі;

3) Розклад доступний для запису пацієнтів.

Main Success Scenario:

1) Адміністратор входить у систему;

2) Обирає лікаря зі списку;

3) Обирає тиждень для планування;

4) Встановлює робочі дні (Mon–Sun);

5) Вказує робочі години (start/end);

6) Система зберігає графік у базі даних;

7) Адміністратор може редагувати окремі дні;

8) Система оновлює розклад у реальному часі.

Extensions:

1) Лікар не обраний – система не дозволяє зберегти графік;
2) Дані не збережено – система показує помилку збереження;
3) Сервер недоступний – повідомлення “Try again later”.

Use Case 5: Додавання та управління тваринами користувача

Scope: Вебзастосунок ветеринарної клініки

Level: User-goal

Primary Actor: Користувач

Stakeholders and Interests:

1) Користувач: додає, редагує та видаляє своїх тварин;
2) Лікар: отримує інформацію про тварину під час прийому;
3) Адміністратор: має доступ до повної історії пацієнтів.

Preconditions:

1) Користувач авторизований у системі;
2) Існує обліковий запис користувача;
3) База даних доступна.

Success Guarantee:

1) Тварина успішно додана або оновлена або видалена;
2) Дані прив’язані до конкретного користувача;
3) Інформація доступна для записів та медичних карток.

Main Success Scenario (додавання тварини):

1) Користувач відкриває особистий кабінет;
2) Переходить у “Мої профіль”;
3) Переходить у розділ “Мої тварини”;
4) Натискає кнопку “Додати тварину”;
5) Система відкриває форму введення даних;
6) Користувач вводить: ім’я тварини, тип, вік, породу;
7) Користувач підтверджує створення;
8) База даних зберігає тварину з прив’язкою до користувача;

9) Оновлений список тварин відображається в інтерфейсі без перезавантаження сторінки.

Extensions:

1) Некоректні дані – система не дозволяє зберегти запис;
2) Сервер недоступний – повідомлення про помилку;
3) Користувач скасував дію – форма закривається без змін.

3.3 Прототипування інтерфейсу користувача

Для візуалізації інтерфейсу вебзастосунку ветиринарної клініки було створено декілька мокапів. Вони є прототипами системи, що створюється і дозволяють перевірити логіку розташування елементів та зручність навігації.

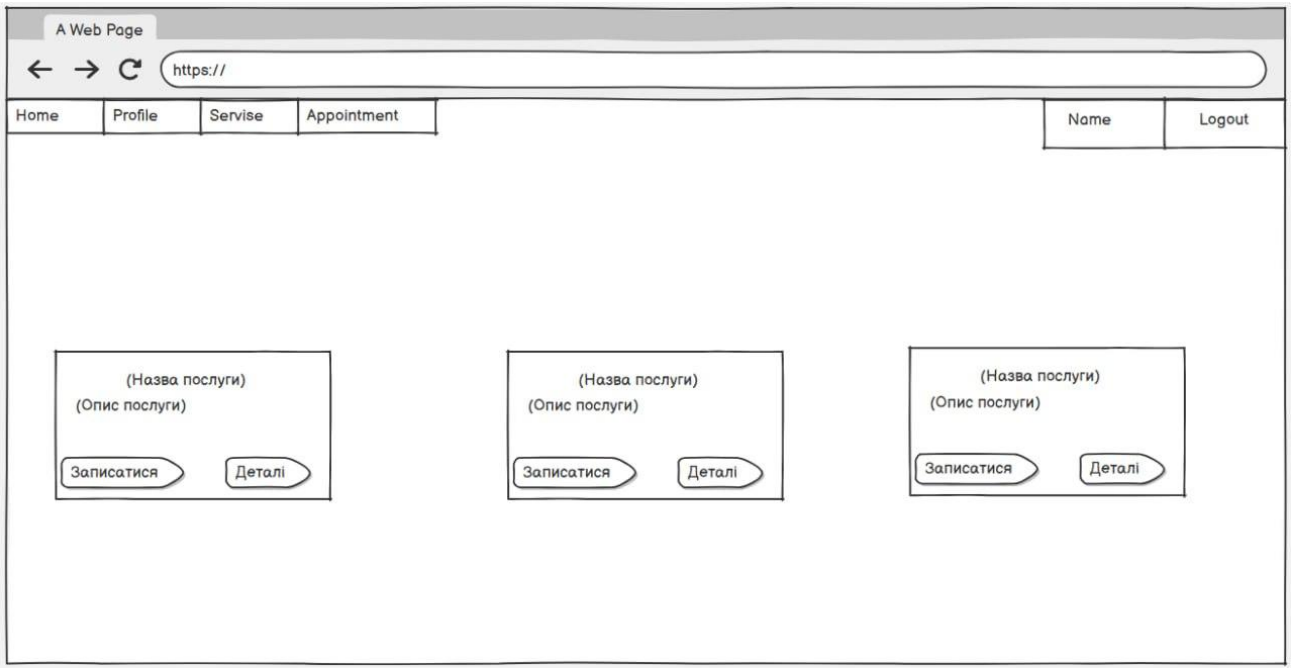

Рисунок 3.4 – Сторінка послуг лікарні

На сторінці Послуги відображається повний перелік усіх послуг ветеринарної клініки. Про кожну послугу відображається її назва, опис послуги, ціна і тривалість процедури. Нижче розташовані кнопки «Записатися» та «Детальніше».

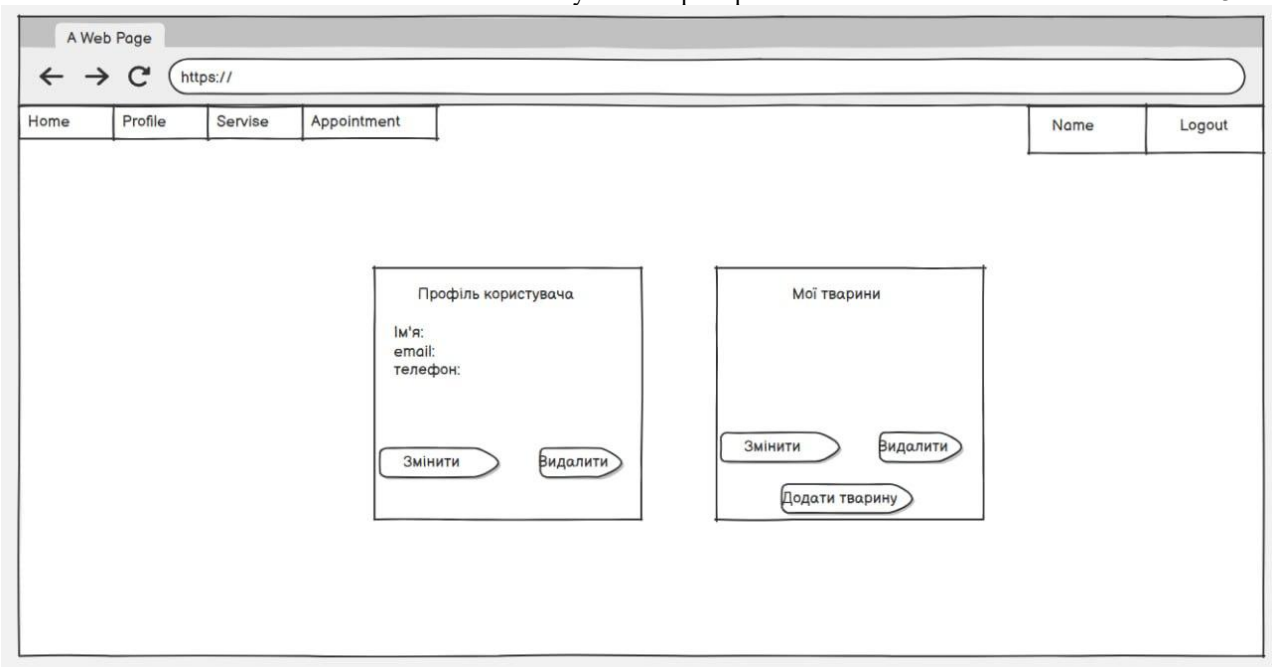

Рисунок 3.5 – Кабінет користувача

Сторінка особистого кабінету користувача (рис. 3.5) призначена для зареєстрованих користувачів. На цій сторінці користувач може переглядати інформацію про самого себе. За потреби користувач може редагувати інформацію натиснувши на кнопку «Змінити» та видалити свій акаунт натиснувши «Видалити».

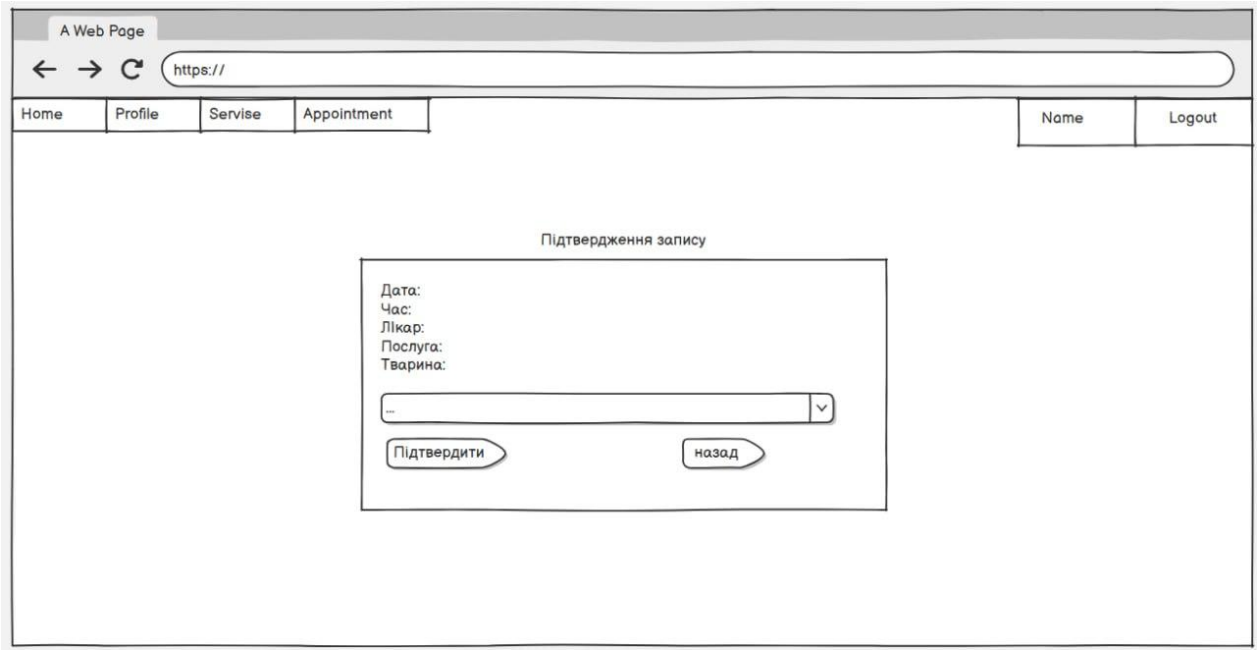

Рисунок 3.6 – Сторінка підтвердження запису на прийом

Сторінка підтвердження запису на прийом (рис 3.6) стає доступною після вибору послуги, лікаря і часу для запису. На цій сторінці відображається інформація,

яку користувач обирав при створенні запису. Нижче розташовано dropdown меню, де відображається список усіх тварин користувача і користувач обирає кого саме записати. Нижче розташовані кнопки «Підтвердити» і «Назад».

Рисунок 3.7 – Сторінка переліку усіх записів користувача

Сторінка переліку усіх записів користувача (рис 3.6) містить інформацію про усі записи на процедури користувача. Для кожного запису відображається така інформація, як дата запису, назва послуги, ім’я лікаря, тварина, що була записана і статус цього запису. Нижче розташована кнопка «Скасувати», яка буде доступна лише для майбутніх записів.

Висновки до розділу 3

У третьому розділі спроєктовано вебзастосунок ветеринарної клініки, що охоплює модель бази даних, діаграму класів, діаграму прецедентів й розгортання, а також створення прототипів ключових інтерфейсів. У результаті було створено загальну схему застосунку, яка враховує чітке розмежування функціоналу відповідно до ролей користувачів: клієнта, лікаря, та адміністратора. Для кожної ролі визначено персональний склад доступних розділів.

У межах проєктування бази даних розроблено логічну модель, візуалізовану у вигляді ER-діаграми. Вона фіксує всі ключові сутності користувачів, тварин, лікарів, послуги, записи на прийом, медичні картки, оалату та сповіщення, а також зв’язки між ними, виражені через зовнішні ключі.

Під час UML-моделювання було створено діаграму розгортання та діаграму прецедентів, які разом дали змогу формалізувати як фізичну архітектуру, так і логіку використання системи. Діаграма розгортання засвідчила трирівневу клієнт-серверну організацію: браузерний фронтенд на Vue.js з AJAX-зверненнями через Axios, серверну частину на PHP та виділений сервер баз даних MS SQL Server. Така архітектура дозволяє незалежно масштабувати вебсервер і сховище, а також модифікувати інтерфейс без порушення роботи логіки. Діаграма прецедентів охопила повний список функцій, доступних різним акторам, від реєстрації та керування профілем тварин до ведення медичних карток, обробки платежів і перегляду статистики. Це підтвердило, що запропонована програмна структура повністю узгоджена із заявленими функціональними вимогами.

Окремим напрямом роботи стало створення мокапів основних екранів інтерфейсу. Створені мокапи дали змогу розглянути інтерфейс майбутнього вебзастосунку, створити зручну навігацію та логіку взаємодії користувача з важливими компонентами: каталогом послуг, особистим кабінетом, формою підтвердження запису та списком усіх прийомів. Прототипи демонструють розмежування можливостей за ролями, наприклад, для адміністратора й лікаря передбачено додаткові органи керування користувачами та записами, тоді як клієнт бачить лише власні дані та особисту інформацію.

Результати отриманні у третьому розділі результати утворюють надійне підґрунтя для подальшої програмної реалізації, тестування та впровадження вебзастосунку ветеринарної клініки, гарантуючи, що всі компоненти системи заздалегідь узгоджені.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ

4.1 Кодування програмного забезпечення

Для реалізації програмного забезпечення використано вебтехнології. Клієнтська частина розроблена з використанням JavaScript фреймворку Vue.js, для забезпечення динамічного інтерфейсу користувача та реактивне оновлення даних. Для організації обміну даними між клієнтською та серверною частинами використовується REST API, який забезпечує взаємодію через HTTP запити у форматі JSON. Зберігання даних реалізовано за допомогою системи управління базами даних Microsoft SQL Server, що забезпечує надійність, цілісність та ефективність роботи з інформацією.

Серверна частина побудована за принципами REST архітектури. Це означає, що кожен ресурс представлений окремим URL endpoint і взаємодія з ним відбувається через HTTP методи:

1) GET: отримання даних;
2) POST: створення нових записів;
3) PUT: оновлення існуючих даних;
4) DELETE: видалення записів.

Усі HTTP-запити обробляються через єдину точку входу в межах кожного модуля, де виконується:

1) перевірка методу запиту;
2) валідація вхідних даних;
3) виклик відповідної бізнес-логіки;
4) формування структурованої відповіді у форматі JSON.

Цей підхід дозволяє створити структуровану та масштабовану систему, у якій кожен компонент відповідає за окрему функціональність. Усі відповіді сервера формуються у форматі JSON, це і забезпечує просту інтеграцію з клієнтською частиною.

Для взаємодії з базою даних використовується розширення PDO (PHP Data Objects), яке забезпечує уніфікований інтерфейс доступу до різних СКБД. У даному

проєкті застосовано драйвер sqlsrv, який забезпечує підключення PHP до Microsoft SQL Server та дозволяє виконувати SQL-запити з використанням підготовлених виразів.

sqlsrv драйвера потрібний для коректної роботи з MS SQL Server, бо він забезпечує підтримку можливостей цієї СКБД, зокрема обробку типів даних SQL Server та оптимізоване виконання запитів. Підключення до бази даних здійснюється через клас Database, який інкапсулює створення PDO з'єднання та забезпечує централізоване управління доступом до БД (рис. 4.1).

```php
$this->conn = new PDO(
    "sqlsrv:Server={$this->server};Database={$this->database}",
    $this->username,
    $this->password
);

$this->conn->setAttribute(
    PDO::ATTR_ERRMODE,
    PDO::ERRMODE_EXCEPTION
);
```

Рисунок 4.1 – Створення PDO з'єднання з MS SQL Server

Особливу увагу приділено безпеці взаємодії з базою даних. Усі SQL-запити реалізовані із застосуванням підготовлених виразів, що і дозволяє запобігти SQL-ін'єкціям. Значення, які отримуються від користувача, не вставляються напряму в SQL-запити, а передаються як параметри, які обробляються драйвером окремо від самої SQL-структури. Це є одним із ключових механізмів захисту серверної частини.

Також реалізовано механізми сесійної автентифікації користувачів через $_SESSION, що дозволяє зберігати стан авторизації після входу в систему. Паролі користувачів не зберігаються у відкритому вигляді, для їх захисту використовується алгоритм bcrypt через функцію password_hash(), а перевірка виконується за допомогою password_verify().

Серверна частина побудована за модульним принципом і поділена на окремі сутності (моделі), кожна з яких відповідає за свою предметну область: користувачі, тварини, записи на прийом, лікарі, послуги, графіки роботи та фінансові операції.

Усі сутності реалізують окрему логічну частину системи та відповідає за зберігання і обробку відповідних даних у базі даних Microsoft SQL Server.

Модель User представляє всіх користувачів вебзастосунку, включаючи клієнтів, лікарів та адміністраторів. Ця модель забезпечує виконання реєстрації, авторизація, отримання інформації про користувача, редагування і видалаення користувача. Вони містить такі поля:

- id: унікальний ідентифікатор користувача;
- name: ім’я користувача;
- email: електронна адреса;
- phone: номер телефону;
- password: захешований пароль;
- role: роль користувача в системі (client, doctor, admin);
- is_verified: статус підтвердження email;
- verification_token: токен підтвердження реєстрації.

Основні методи цієї моделі:

- getById($id): це метод призначений для отримання повної інформації про користувчача;
- update($data): метод використовується для новлення та редагування інформації про користувача. Цей метод дозволяє змінювати ім’я, пошут, телефон і пароль користувача;
- delete($id): цей метод потрібен для видалення користувача з системи;
- register($data): метод використовується для реєстрації користувача. Перед створенням профілю користувача відбувається перевірка унікальності пошти. Пароль користувача хешується алгоритмом bcrypt;
- createUserByAdmin($data): цей метод використовується для створення нових користувачів адміністратором;
- createDoctor($data): метод для створення нових лікарів. Крім даних про звичайних користувачів, цей метод зберігає професійну інформацію лікаря;
- getAllUsers(): цей метод повертає список усіх зареєстрованих користувачів у системі;

– login($email, $password): метод, що здійснює аутентифікацію користувача. Цей метод перевіряє наявність користувача в системі, коректність введених даних.

Модель Pet представляє інформації про домашніх тварин клієнтів клініки. Вона забезпечує створення, редагування і видалення тварин. Поля цієї моделі:

– id: унікальний ідентифікатор тварини;
– name: ім’я тварини;
– user_id: ідентифікатор власника;
– type: тип тварини;
– breed: порода;
– age: вік;
– notes: додаткова інформація.

Основні методи цієї моделі:

– create($data): цей метод призначений для створення запису про нову тварину;
– getByUserId($userId): метод призначений для отримання списку тварин, що належать певному користувачу;
– update($data): метод забезпечує оновлення редагування інформації про тварину;
– delete($id): цей метод використовується для видалення запису про тварину з бази даних.

Модель Appointment відображає дані про записи на прийому користувачі. Вона забезпечує створення і скасування записів, а також перевірку доступності лікаря. Містить поля:

– id: унікальний id запису;
– user_id: клієнт, який записався;
– pet_id: тварина, яку записали;
– doctor_id: лікар, до якого записали;
– service_id: обрана послуга;
– appointment_datetime: дата і час прийому;

- status: статус запису;
- created_at: коли запис було створено.

Основні методи цієї моделі:

- create($data): цей метод використовується для створення запису на прийом. Перед створенням перевіряється наявність обов’язкових даних та доступність лікаря у обраний час;
- cancel($id): цей метод зміняю статус запису на скасовано;
- getByDate($date, $doctorId = null): метод дозволяє отримати записи на певну дату з можливіст’ю фільтрації по лікарю;
- isDoctorBusy($doctorId, $datetime, $serviceId): цей метод перевіряє доступність лікаря у вибраний проміжок часу. Метод отримує тривалість послуги, обчислює часовий інтервал і перевіряє з існуючими записами;
- getAll(): цей метод повертає список усіх записів у системі. Також містить додаткову інформацію для користувачу, таку як назва послуги та ім’я лікаря;
- getByUser($user_id): метод отримує усі записи, які прив’язані до певного користувача.

Модель Service представляє інформацію про усі послуги ветеринарної клініки. Її поля:

- id: унікальний id послуги;
- name: назва послуги;
- description: опис послуги;
- price: вартість;
- duration: тривалість виконання.

Основні методи моделі:

- create($data): метод для створення нової послуги для клініки;
- update($data): метод для оновлення або редагування інформації про вже існуючі послуги;
- delete($id): метод для видалення послувги;
- getDuration($id): цей метод використовується при розрахунках доступності лікаря при створенні запису на прийом;

– getDoctorsMap(): цей метод створює відповідність між лікарями і послугами, які вони виконують.

Модель Doctor демонструє дані про лікарів клініки, вона містить такі поля:

– user_id: зв’язок із користувачем;
– specialization: спеціалізація лікаря;
– experience: досвід роботи;
– description: опис лікаря.

Основні методи моделі:

– getById($id): отримання повної інформації по певному лікарю;
– getAll(): отримання переліку усіх лікарів.

Модель Record показує інформацію про призначення лікаря після прийому. Поля цієї моделі:

– id: унікальний ідентифікатор запису;
– appointment_id: зв’язок із прийомом;
– diagnosis: діагноз;
– treatment: лікування;
– notes: додаткові примітки;
– files: прикріплені файли.

Основні методи моделі:

– create($data): метод для створення призначення лікарем;
– findByAppointment($appointment_id): отримання медичного запису до конкретного прийому;
– delete($id): метод для видалення медичного запису.

Модель Schedule відображає інформацію про робочий графік лвкарів. Її поля:

– id: ідентифікатор запису розкладу;
– doctor_id: лікар;
– work_date: дата роботи;
– start_time: початок робочого дня;
– end_time: кінець робочого дня.

Основні методи моделі:

- getByDoctor($doctorId): метод для перегляду графіку роботи певного лікаря;
- getDoctorBusySchedule($doctorId, $date): цей метод використовується для перегляду записів до лікаря на певний день;
- create($data): метод для створення розкладу роботи лікаря на тиждень;
- getByDoctorsAndDate($doctorIds, $date): метод для отримання графіків роботи декількох лікарів;
- update($data): метод для оновлення графіку роботи лікаря.

Модель Notification відображає інформацію про нагадування, що приходять лікарю. Її поля:

- id: унікальний id сповіщення;
- user_id: отримувач повідомлення;
- from_user_id: ініціатор повідомлення;
- type: тип сповіщення;
- title: заголовок;
- message: текст повідомлення;
- appointment_id: зв’язок із записом на прийом;
- record_id: зв’язок із медичним записом;
- is_read: статус прочитання;
- created_at: дата створення.

Основні методи моделі:

- create($data): цей метод виконує вставку у таблицю Notification і за замовченням значення у поля is_read=0;
- getUnreadCount($user_id): метод повертає кількість непрочитаних повідомлень користувача;
- markAsRead($id): цей метод позначає повідомлення як причитене, встановлюючи значення is_read=1.

Серверна частина вебзастосунку реалізована у REST API, що забезпечує взаємодію між клієнтською частиною та базою даних Microsoft SQL Server. Усі

функціональні операції системи винесені в окремі маршрути, які обробляють HTTP-запити та повертають відповіді у форматі JSON.

Маршрут /register використовується для створення нового користувача. Під час виконання перевіряється унікальність email, виконується хешування пароля та створюється новий запис у таблиці Users.

```php
if ($method === 'POST' && $_GET['action'] === 'register') {

    $data = json_decode(file_get_contents("php://input"), true);

    if (!isset($data['name'], $data['email'], $data['password'])) {
        Response::error("Missing fields");
    }

    $result = $userModel->register($data);

    if (($result['code'] ?? null) === 'NOT_VERIFIED_EXISTS') {

    $stmt = $db->prepare("
        SELECT verification_token,email
        FROM Users
        WHERE id=?
    ");

    $stmt->execute([$result['user_id']]);

    $u = $stmt->fetch(PDO::FETCH_ASSOC);

    sendVerificationEmail(
        $u['email'],
        $u['verification_token']
    );
```

Рисунок 4.2 – Маршрут /register

Маршрут /login виконує перевірку даних користувача. Після перевірки email і пароля створюється сесія користувача.

```php
if ($method === 'POST' && $_GET['action'] === 'login') {

    $data = json_decode(file_get_contents("php://input"), true);

    if (!isset($data['email']) || !isset($data['password'])) {
        Response::error("Missing credentials");
    }

    $user = $userModel->login(
        $data['email'],
        $data['password']
    );
    if (isset($user['error'])) {

        if ($user['error'] === 'EMAIL_NOT_VERIFIED') {
            Response::error("EMAIL_NOT_VERIFIED");
        }

        Response::error("INVALID_CREDENTIALS");
    }
    $_SESSION['user'] = $user;

    Response::success($user, "Login successful");
}
```

Рисунок 4.3 – Маршрут /login

Маршрут /pets реалізує повний функціонал CRUD для сутності Pet та забезпечує взаємодію користувача з його домашніми тваринами.

Метод GET (рис. 4.4) використовується для отримання всіх тварин, які належать певному користувачу.

```php
if ($method === 'GET') {

    if (!isset($_GET['user_id'])) {
        Response::error("user_id is required");
    }

    $data = $controller->getByUser($_GET['user_id']);
    Response::success($data);

}
```

Рисунок 4.4 – Метод GET

Метод POST (рис. 4.5) використовується для додавання тварин до користувача.

```php
if ($method === 'POST') {

    $data = json_decode(file_get_contents("php://input"), true);

    if (!$data) {
        Response::error("No JSON body");
    }

    $result = $controller->create($data);

    if (!$result) {
        Response::error("Create failed");
    }

    Response::success($result, "Created");
}
```

Рисунок 4.5 – Метод POST

Метод PUT (рис. 4.6) використовується для редагування інформації про вже існуючих тварин.

.

```php
if ($method === 'PUT') {

    $data = json_decode(file_get_contents("php://input"), true);

    $result = $controller->update($data);

    Response::success($result, "Updated");
}
```

Рисунок 4.6 – Метод PUT

Метод DELETE (рис. 4.7) використовується для видалення тварин.

```php
if ($method === 'DELETE') {

    $data = json_decode(file_get_contents("php://input"), true);

    if (!isset($data['id'])) {
        Response::error("id is required");
    }

    $result = $controller->delete($data['id']);

    if (!$result) {
        Response::error("Delete failed");
    }

    Response::success(null, "Deleted");
}
```

Рисунок 4.7 – Метод DELETE

API користувачів забезпечує управління обліковими записами та дозволяє виконувати операції перегляду, оновлення та видалення користувачів.

Метод GET (рис. 4.8) використовується для отримання інформації про конкретного користувача.

```php
if ($method === 'GET') {

    if (!isset($_GET['id'])) {
        Response::error("id is required");
    }

    $data = $controller->getById($_GET['id']);
    Response::success($data);
}
```

Рисунок 4.8 – Метод GET

Метод DELETE (рис. 4.9)використовується для видалення користувача з системи.

```php
if ($method === 'DELETE') {

    $data = json_decode(file_get_contents("php://input"), true);

    if (!isset($data['id'])) {
        Response::error("id is required");
    }

    $result = $controller->delete($data['id']);

    if (!$result) {
        Response::error("Delete failed");
    }

    Response::success(null, "Deleted");
}
```

Рисунок 4.9 – Метод DELETE

Маршрут /appointments?user_id=x&full=x (рис. 4.10) використовується для отримання розширеної інформації про записи користувача з використанням JOIN.

```php
if ($method === "GET" && isset($_GET['user_id']) && isset($_GET['full'])) {

    $data = $controller->getFullByUser($_GET['user_id']);

    Response::success($data);
    exit;
}
```

Рисунок 4.10 – Метод GET для отримання розширеної інформації про записи

Маршрут /appointments?busy=1&date=x (рис. 4.11) використовується для отримання списку зайнятих часових інтервалів лікаря на певну дату.

```php
if ($method === "GET" && isset($_GET['busy']) && isset($_GET['date'])) {

    $data = $controller->getBusyIntervalsByDate($_GET['date']);

    Response::success($data);
    exit;
}
```

Рисунок 4.11 – Метод GET для отримання інформації про зайняті інтервали

Метод GET (рис. 4.12) призначений для отримання зайнятості лікаря.

```php
if ($method === "GET") {

    $doctorId = $_GET['doctor_id'] ?? null;
    $date = $_GET['date'] ?? null;
    $mode = $_GET['mode'] ?? "user";


    if ($doctorId && $date && $mode === "doctor") {

        $data = $controller->getDoctorBusySchedule($doctorId, $date);

        echo json_encode([
            "success" => true,
            "message" => "OK",
            "data" => $data
        ]);
        exit;
    }
```

Рисунок 4.12 – Метод GET для отримання зайнятості лікаря

Метод POST (рис 4.13) призначений для створення розкладу лікаря.

```php
if ($method === "POST") {

    $data = $_POST;

    if (empty($data)) {
        $data = json_decode(file_get_contents("php://input"), true);
    }

    if (!isset($data["doctor_id"]) || !isset($data["week_start"])) {
        echo json_encode([
            "success" => false,
            "message" => "doctor_id and week_start required"
        ]);
        exit;
    }
```

Рисунок 4.12 – Метод GET для заповнення розкладу лікаря

Клієнтська частина вебзастосунку реалізована із використанням сучасного JavaScript-фреймворку Vue.js, який забезпечує компонентну архітектуру, реактивне оновлення даних та зручну інтеграцію з серверною частиною через REST API.

Однією з головних функцій клієнтської частини є механізм авторизації користувачів (рис. 4.13). У разі успішної авторизації сервер повертає дані користувача, після цього користувач переходить на сторінку власного профілю.

```
const res = await axios.post(
  "http://localhost:8000/api/auth.php?action=login",
  {
    email: email.value,
    password: password.value
  }
`
```

Рисунок 4.13 – Реалізація авторизації

Реєстрація нового користувача реалізовано через іншу форму. Після успішної реєстрації користувачу відображається повідомлення про те, що користувачу на вказану пошту надійде лист і там потрібно підтвердити пошту.

```
const res = await axios.post(
  'http://localhost:8000/api/auth.php?action=register',
  {
    name: name.value,
    email: email.value,
    password: password.value,
    phone: phone.value
  }
)
```

Рисунок 4.14 – Реалізація реєстрації

Одним із ключових модулів системи є управління даними тварин користувача. Над сутністю Pet реалізовано повний CRUD функції.

```
if (isEditMode.value) {
  await axios.put("http://localhost:8000/api/pets.php", petForm.value)
} else {
  await axios.post("http://localhost:8000/api/pets.php", {
    user_id: userId.value,
    ...petForm.value
  })
}
```

Рисунок 4.15 – Додавання та редагування тварини

```
confirmAction.value = async () => {
  await axios.delete("http://localhost:8000/api/pets.php", {
    data: { id: petId }
  })
```

Рисунок 4.15 – Видалення тварини

Також у клієнтській частині реалізовано модуль для керуванням медичними послугами. Цей функціонал використовується адміністратором і дозволяє виконувати CRUD операції над послугами.

```
try {
  const res = await axios.get("http://localhost:8000/api/services.php")
  services.value = res.data.data
```

Рисунок 4.16 – Отримання переліку послуг

Для додавання нової послуги використовується модальне вікно, де адміністратор має вводити дані про послугу (рис. 4.17).

```
const openCreate = () => {
  isEdit.value = false

  form.value = {
    id: null,
    name: "",
    description: "",
    price: "",
    duration: "",
    doctors: []
  }

  showModal.value = true
}
```

Рисунок 4.17 – Форма створення послуги

```
if (isEdit.value) {
  await axios.put("http://localhost:8000/api/services.php", payload)
} else {
  await axios.post("http://localhost:8000/api/services.php", payload)
}
```

Рисунок 4.18 – Збереження послуги

Модуль керування записами користувача забезпечує перегляд історії візитів, отримання медичних рекомендацій, скасування записів та завантаження прикріплених файлів.

Для кожного запису користувач може переглядати рекомендації лікаря (рис. 4.19).

```
const res = await axios.get(
  "http://localhost:8000/api/records.php",
  {
    params: {
      appointment_id: appointment.id
    }
  }
)

selectedRecord.value =
  res.data.data?.[0] || null

showRecordModal.value = true
```

Рисунок 4.19 – Отримання медичного запису

Також в системі передбачено можливість завантаження файлів, з назначень лікаря (рис. 4.20).

```
window.open(
`http://localhost:8000/api/file.php?name=${encodeURIComponent(file)}`,
"_blank"
)
```

Рисунок 4.20 – Відкриття файлу

4.2 Тестування програмного забезпечення

Для перевірки коректності роботи програми і відповідності реалізованого функціоналу до вимог технічного завдання було проведено тестування основних модулів. Основною метою тестування є перевірка стабільної роботи вебзастосунку.

Таблиця 4.1 – Тест-кейси вебзастосунку

№	Назва тесту	Попередні умови	Кроки тестування	Очікуваний результат
1	Реєстрація нового користувача	Відкрита сторінка реєстрації	1) Ввести ім’я, пошту, телефон і пароль; 2) Натиснути «Зареєструватися».	Створення нового запису в таблиці Users; повідомлення про успішну реєстрацію; надходження листа на email.

Продовження таблиці 4.1

2	Вхід користувача в систему	Існує зареєстрований користувач	1) Ввести пошту і пароль; 2) Натиснути «Увійти».	Отримання даних користувача та збереження сесії; перенаправлення в профіль.
3	Відображення профілю користувача	Користувач авторизований	1) Відкрити сторінку «Мій профіль»	Відображаються дані користувача.
4	Редагування даних користувача	Користувач перебуває у профілі	1) Натиснути «Змінити»; 2) Внести зміни; 3) 3) Натиснути «Зберегти».	Оновлення запису в таблиці Users; відображення оновлених даних без перезавантаження сторінки.
5	Видалення даних користувача	Користувач авторизований	1) Натиснути «Видалити акаунт»; 2) Підтвердити дію.	Видалення запису з таблиці Users; очищення сесії; перенаправлення на сторінку реєстрації.
6	Додавання нової тварини	Користувач зареєстрований	1) Натиснути «Додати тварину»; 2) Заповнити форму; 3) Зберегти.	Створення запису в таблиці Pets; відображення нової тварини в списку.
7	Видалення тварини	У користувача є тварина	1) Натиснути «Змінити» біля тварини; 2) Внести зміни; 3) Зберегти.	Видалення запису з Pets; оновлення списку тварин.

Продовження таблиці 4.1

8	Редагування інформації про тварину	У користувача є тварина	1) Натиснути «Змінити» біля тварини; 2) Внести зміни; 3) Зберегти.	Оновлення даних у таблиці Pets; оновлення UI без перезавантаження.
9	Перегляд списку записів	Користувач має записані прийоми	1) Відкрити сторінку «Мої записи».	Відображається список прийомів із лікарем, датою, послугою та статусом
10	Скасування запису на прийом	Існує активний запис	1) Натиснути «Скасувати» біля запису.	Зміна статусу на «cancelled»; оновлення списку без перезавантаження
11	Перегляд назначення лікаря	Існує запис із лікарським висновком	1) Натиснути «Переглянути назаначення».	Відображення модального вікна з діагнозом, лікуванням та нотатками
12	Завантаження файлів із назначення	У записі є прикріплені файли	1) Натиснути на файл.	Відкриття файлу через API /file.php у новій вкладці.
13	Перегляд розкладу лікаря	Лікар авторизований	1) Відкрити розклад; 2) Перемикати тижні.	Відображення тижневого графіка з прийомами та робочими днями.

Продовження таблиці 4.1

14	Створення розкладу на тиждень	Лікар авторизований	1) Обрати шаблон; 2) Натиснути «Застосувати».	Створення записів у таблиці Schedule; оновлення календаря.
15	Редагування робочого дня	Існує запис у розкладі	1) Вибрати день; 2) Змінити час; 3) Зберегти	Оновлення запису в Schedule; зміна відображення графіка.
16	Перегляд послуг	Користувач авторизований	1) Відкрити «Наші послуги».	Відображення списку послуг із ціною та тривалістю.
17	Створення нової послуги	Користувач має роль адміністратора	1) Натиснути «Додати послугу»; 2) Заповнити форму.	Додавання запису в Services; оновлення списку.
18	Прив’язка лікарів до послуги	Адмін відкрив редагування послуги	1) Обрати лікарів; 2) Зберегти.	Оновлення зв’язку ServiceDoctors у базі даних.
19	Перегляд лікарів	Користувач авторизований	1) Відкрити сторінку «Лікарі».	Відображається список лікарів.

4.3 Керівництво користувача

Створена система є вебзастосунком для взаємодії користувача, лікаря і адміністратора. На рисунку 4.21 продемонстровано головна сторінка сайту. Для незареєстрованого або неавторизованого користувача на навігаційній панелі відображаються кнопки «Логін» та «Реєстрація». Нижче розташована інформація про саму клініку.

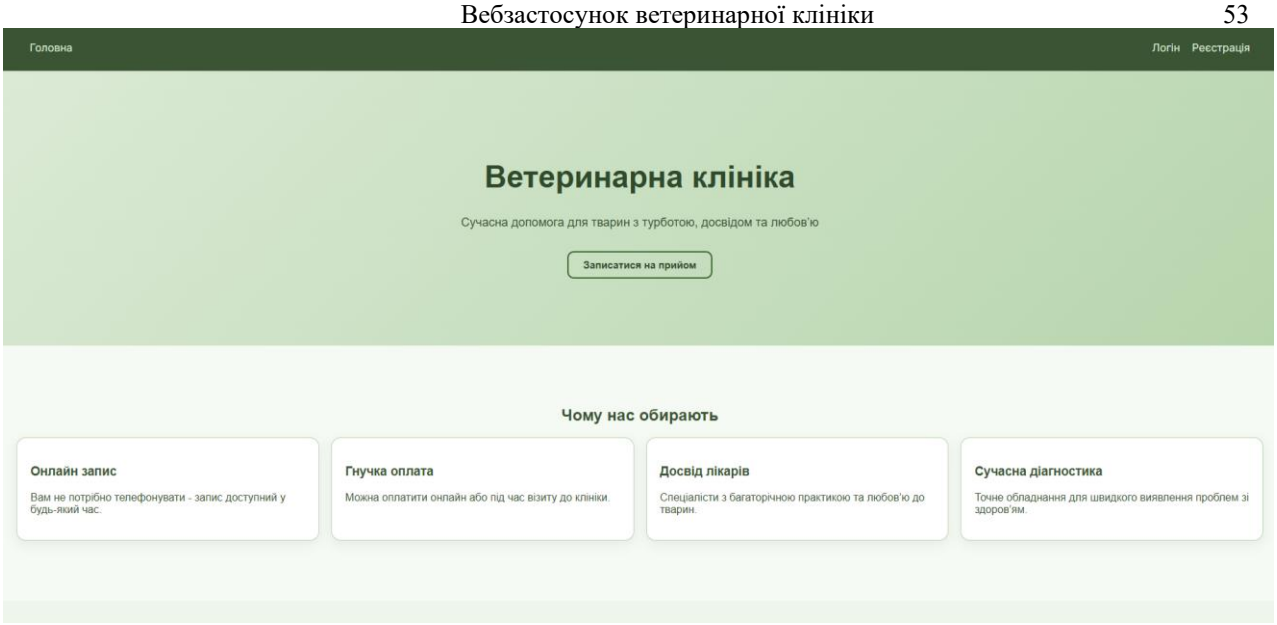

Рисунок 4.21 – Головна сторінка сайту

На рисунку 4.22 показана форма авторизації вже зареєстрованого користувача. На цій сторінці є два поля для вводу пошти і паролю, і кнопка для входу.

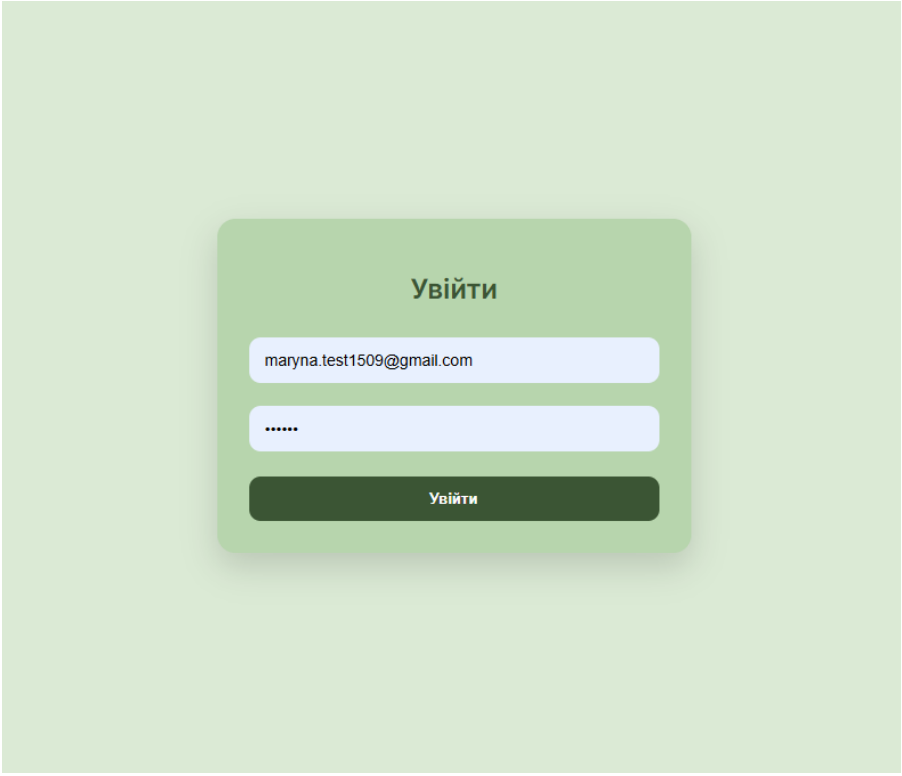

Рисунок 4.22 – Сторінка авторизації

Сторінка реєстрації (рис. 4.23) призначена для реєстрації нового користувача. В цю форму користувач вводить такі дані як: ім’я, пошта, телефон і пароль. Після

натискання кнопки «Зареєструватися» система перевіряє унікальність пошти. У разі успішної реєстрації користувачу надійде лист на пошту для її підтвердження.

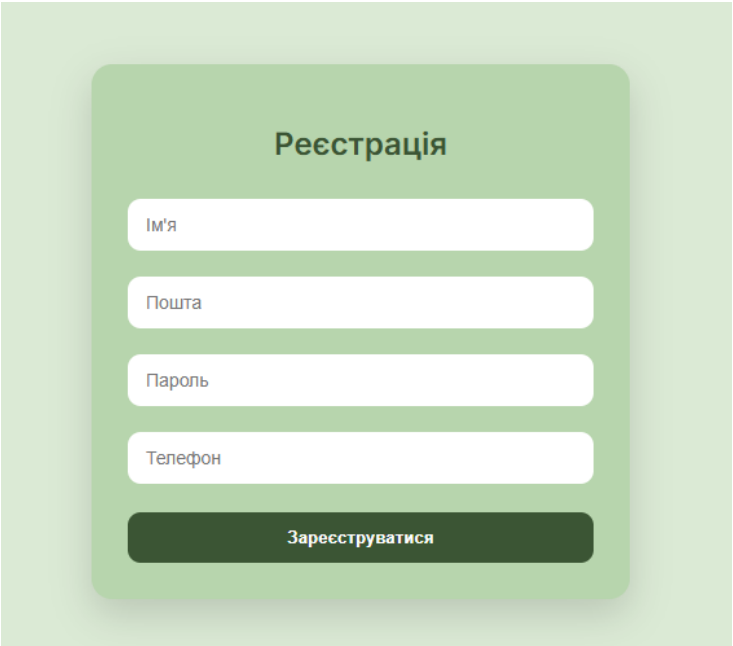

Рисунок 4.23 – Сторінка реєстрації

Після підтвердження пошти користувач перейде на сторінку «Мій профіль» (рис. 4.24). на цій сторінці відображається особиста інформація про користувача, така як: ім’я користувача, пошта та номер телефону. Нижче розташовані кнопки «Змінити» та «Видалити». Справа розташовано блок «Мої тварини», де відображається інформація про усіх тварин користувача.

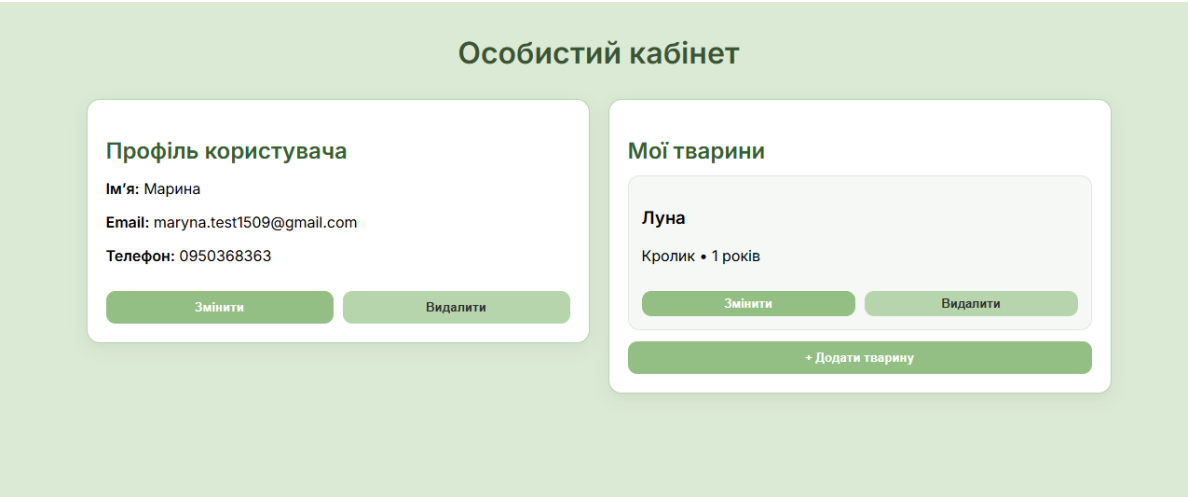

Рисунок 4.24 – Особистий кабінет

На сторінці «Наші послуги» знаходяться усі послуги клініки (рис. 4.25). Кожна послуга відображається як окремий блок і всередині кожного блоку відображається

назва послуги, її скорочений опис, ціна послуги і її тривалість. Нижче розташовані кнопки «Записатися» і «Детальніше».

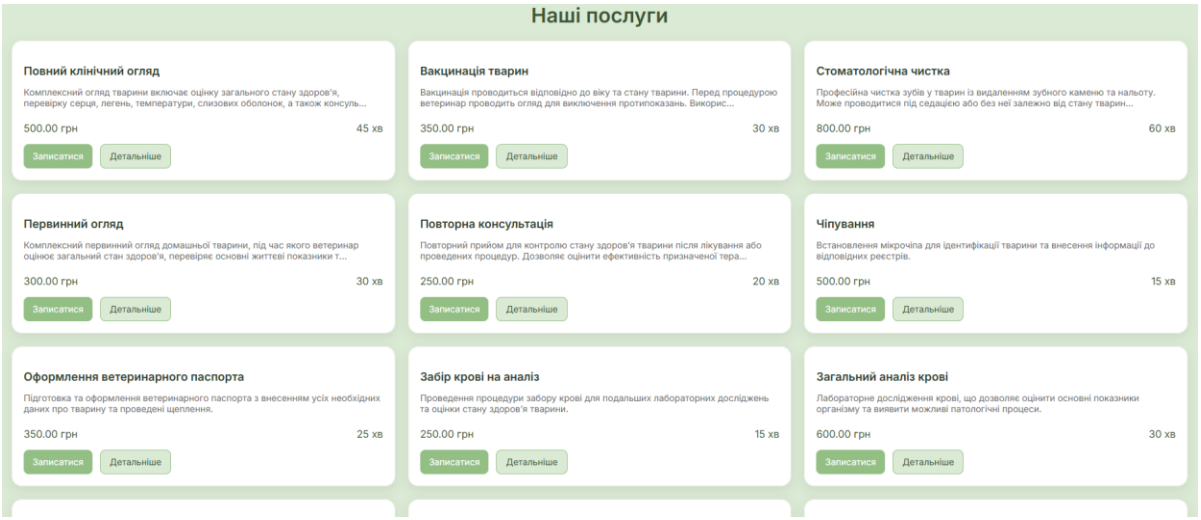

Рисунок 4.25 – Сторінка «Наші послуги»

При натисканні на кнопку «Записатися» користувач переходить на сторінку «Плануйте консультацію» де відображається календар на теперішній місяць (рис. 4.26). Над календарем розташовані кнопки для зміни місяцю для запису. Правіше розташовані часові слоти, які фільтруються в залежності від того, якого саме лікаря обирає користувач, під переліком часових слотів знаходиться кнопка «Записатися».

Рисунок 4.26 – Сторінка «Плануйте консультацію»

Після натискання на кнопку «Записатися» користувач переходить на сторінку «Підтвердження запису». На цю сторінку підвантажуються дані зі сторінки «Плануйте консультацію». На сторінці відображається дата, час, ім’я лікаря і

послуга. Нижче розташовано меню, де користувач має обрати яку саме з своїх тварин він хоче записати. Далі розташовані кнопки «Підтвердження запису» і «Назад».

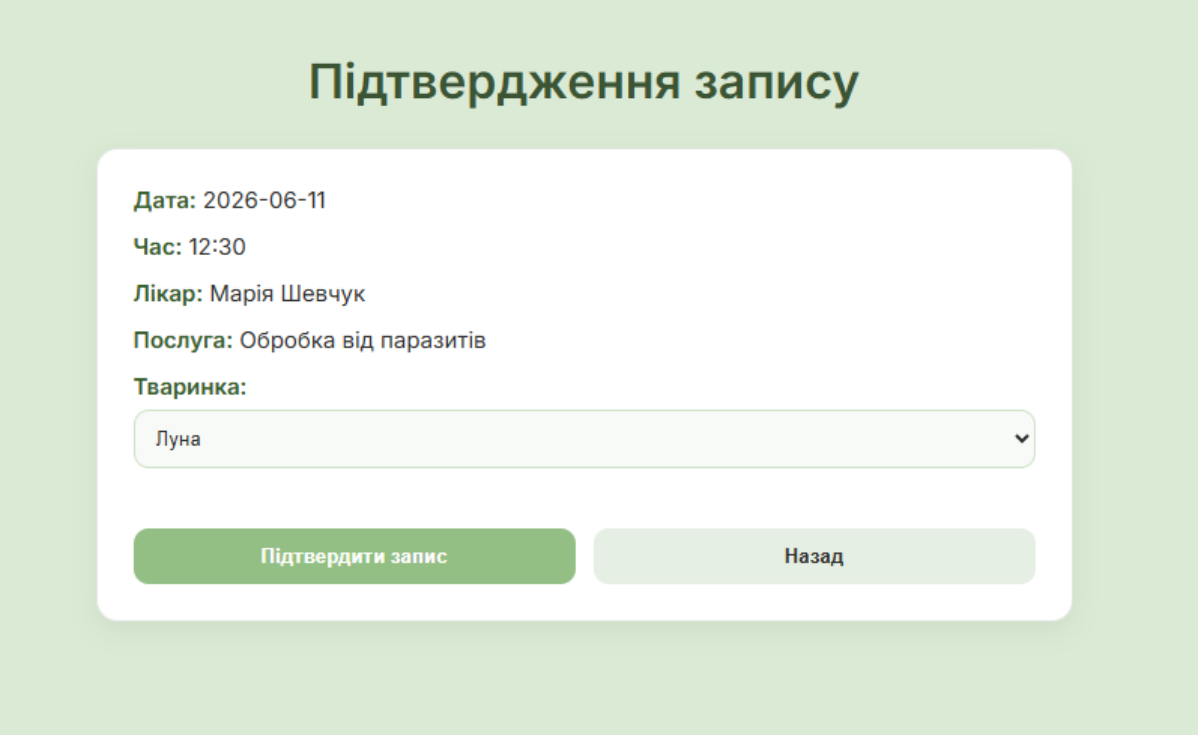

Рисунок 4.27 – Сторінка «Підтвердження запису»

Після підтвердження запису, запис на консультацію з’явиться у вкладці «Мої записи» (рис. 4.28). В цьому блоці відображається дата, послуга, лікар, тварина і статус. Якщо запис вже був завершений, то з’явиться кнопка «Назначення лікаря». Якщо запис ще не відбувся, то користувач має кнопку «Скасувати».

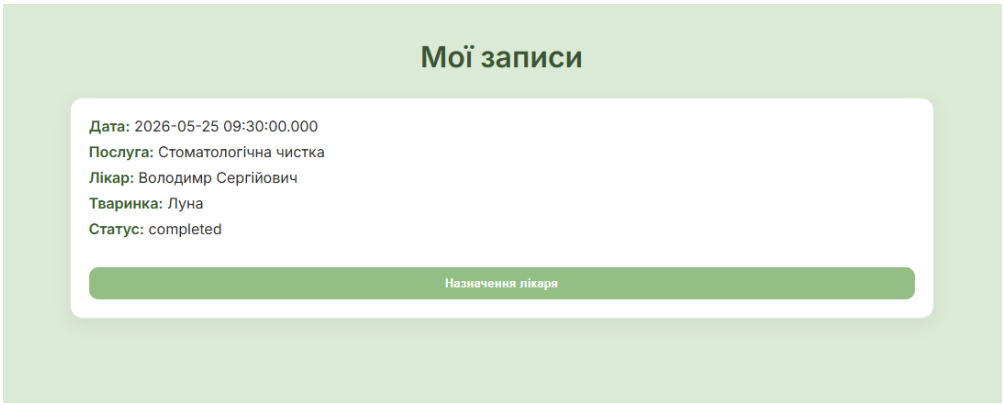

Рисунок 4.28 – Сторінка «Підтвердження запису»

У лікаря є особлива вкладка «Розклад» (рис. 4.29). На цій сторінці у лікаря відображається інформація про усі записи на найближчий тиждень. Лікар може дивитися розклад, як на цей тиждень, так і на попередній і на тиждень що вже минув.

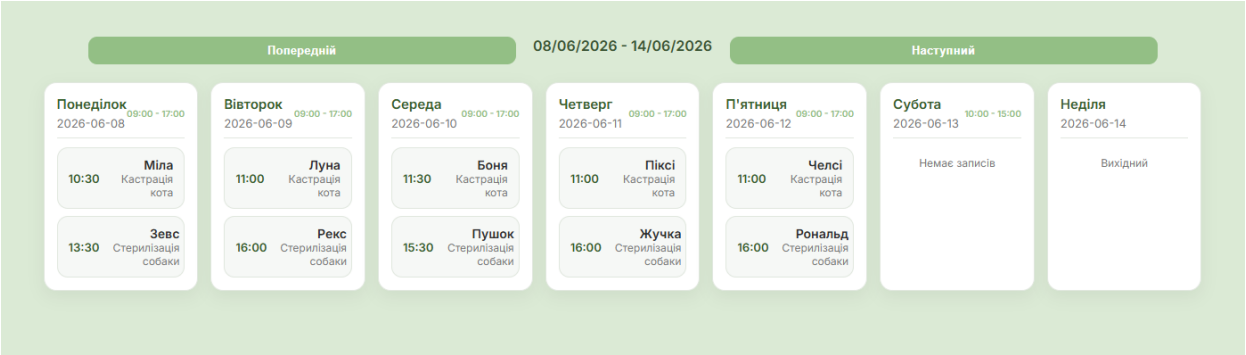

Рисунок 4.29 – Сторінка «Розклад» у лікаря

При натисканні на запис, одразу відкривається модальне вікно (рис. 4.30). Там відображається повна інформація про запис: час, хазяїн, тварина, послуга і дата створення. Справа розташовано блок «Назначення», в цій частині лікар може залишити діагноз, лікування, нотатки і також додати туди файл, за потреби

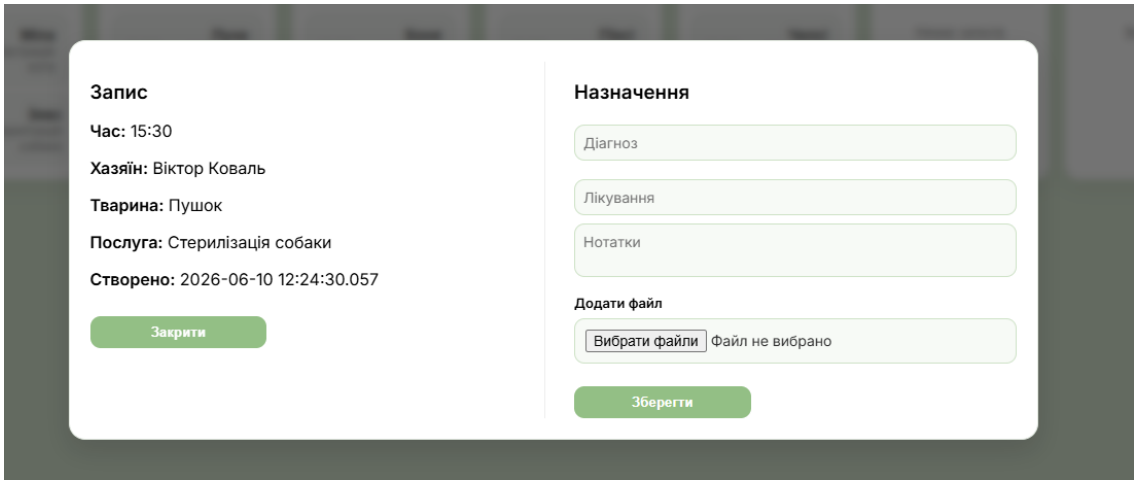

Рисунок 4.30 – Сторінка «Назначення» у лікаря

Також у лікаря є вкладка «Пацієнти» (рис. 4.31). Саме на цій вкладці лікар може знайти усіх тварин одного користувача, а також переглянути повну історію хвороб і назначень для лікування. Для кожної тварини відображається її ім’я, її хазяїн, тип, породи і нотатки. Справа розташована кнопка «Відкрити».

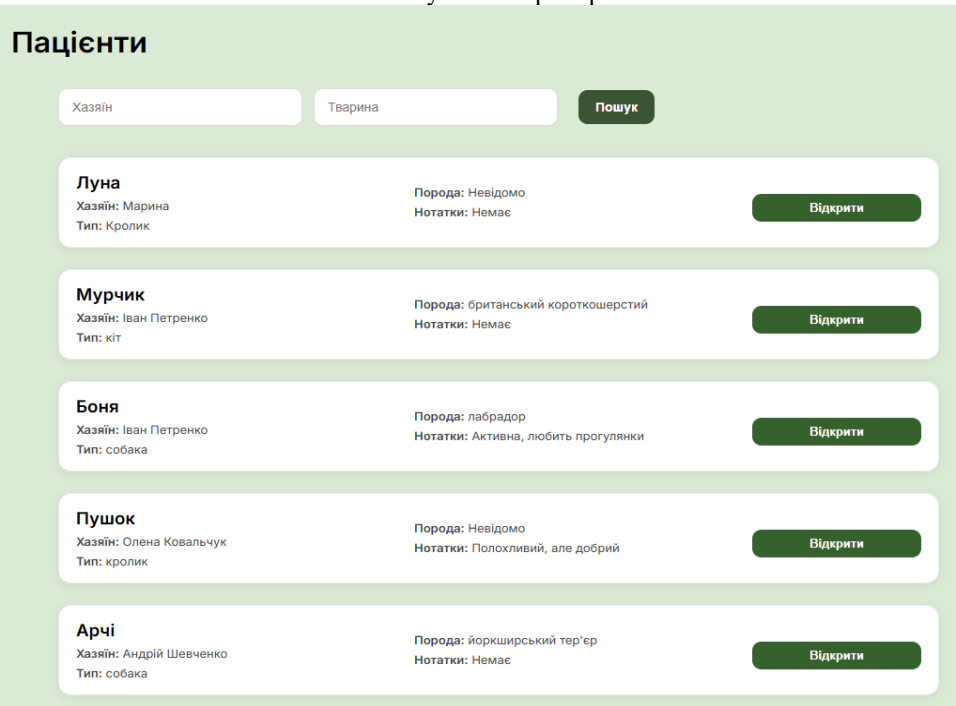

Рисунок 4.31 – Сторінка «Пацієнти» у лікаря

При натисканні на кнопку «Відкрити» відкривається модальне вікно з відображенням історії записів певної тварини і назначення лікаря до кожного з записів (рис. 4.32).

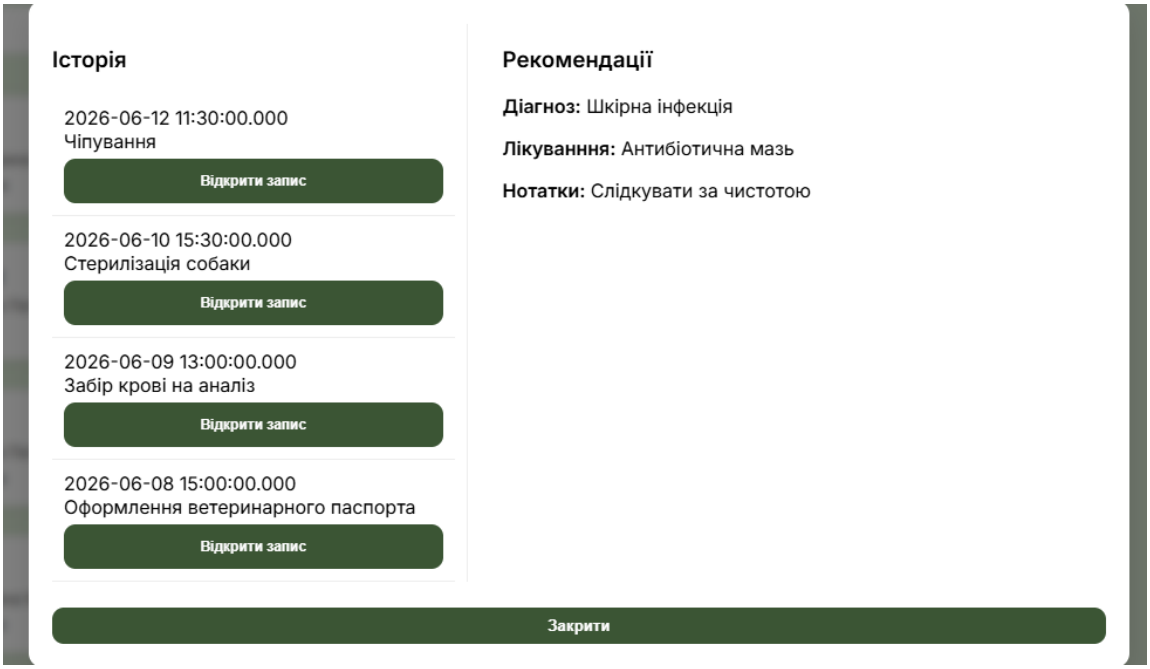

Рисунок 4.32 – Історія лікування пацієнтів

Висновки до розділу 4

В четвертому розділі було розглянуто програмну реалізацію вебзастосунку ветеринарної клініки. Було охоплено розробку серверної частини де реалізовано взаємодію між ними за допомогою REST API та забезпечено обмін даними у форматі JSON.

На серверній стороні реалізовано програмні модулі для роботи з користувачами, домашніми тваринами, лікарями, ветеринарними послугами, розкладом роботи спеціалістів, записами на прийом та медичними картками. Для взаємодії з базою даних використано SQL Server та технологію PDO, що забезпечує надійне та безпечне виконання операцій із даними.

Клієнтська частина розроблена з використанням фреймворку Vue.js. Реалізовано функціонал реєстрації та авторизації користувачів, управління профілем, ведення обліку домашніх тварин, створення та скасування записів на прийом.

Для підтвердження працездатності програмного забезпечення було проведено функціональне тестування основних модулів системи. Результати тестування показали коректну роботу реалізованого функціоналу та відповідність отриманого програмного продукту поставленим вимогам.

Крім програмної реалізації, було підготовлено керівництво користувача, що містить опис основних функцій системи та порядок роботи з ними. Наявність цього керівництва спрощує процес освоєння програмного продукту.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи проведено комплексний аналіз предметної області ветеринарних клінік, досліджено сучасний стан автоматизації та обґрунтовано вибір технологій для створення вебзастосунку. На основі отриманих даних сформульовано повну специфікацію вимог до програмного продукту.

Під час аналізу існуючих рішень (Veles Vet Clinic та ЕКСВЕТ) виявлено, що навіть функціонально розвинені системи не забезпечують повноцінної підтримки онлайн запису, оплати онлайн, управління персоналом та розмежування прав доступу. Це підтвердило актуальність розроблення нового вебзастосунку, здатного усунути зазначені недоліки.

Визначено перелік із 20 основних функцій майбутньої системи, серед яких: реєстрація та авторизація з розподілом ролей (адміністратор, лікар, клієнт), ведення медичних карток тварин, онлайн-запис на прийом з інтерактивним календарем, автоматичні нагадування, фінансовий модуль з онлайн-оплатою, генерація статистичних звітів. Для кожної ролі детально описано ключові варіанти використання (Use Cases), що охоплюють основні робочі сценарії клініки.

Обґрунтовано вибір технологічного стеку: серверна частина на PHP із використанням MS SQL Server як надійної реляційної СКБД промислового рівня, клієнтська частина – на JavaScript із фреймворком Vue.js та бібліотекою Axios для побудови динамічного односторінкового інтерфейсу. Доведено, що таке поєднання забезпечує необхідний баланс продуктивності, безпеки, масштабованості та зручності супроводу.

Сформульовано комплекс функціональних і нефункціональних вимог, який визначає повну поведінку системи, а також вимоги до безпеки (HTTPS, хешування паролів, захист від ін'єкцій), надійності, продуктивності, масштабованості та сумісності із сучасними браузерами й серверними платформами.

У третьому розділі роботи спроєктовано загальну структуру вебзастосунку з урахуванням розмежування прав доступу для клієнта, лікаря, та адміністратора. Розроблено ER-діаграму бази даних, яка фіксує всі ключові сутності (користувачі,

тварини, лікарі, послуги, записи, медичні картки, платежі, сповіщення) та зв'язки між ними. Побудовано UML-діаграми розгортання та прецедентів, що формалізують фізичну архітектуру (трирівнева клієнт-серверна модель) та логіку використання системи відповідно до ролей. Крім того, створено низку прототипів інтерфейсу (каталог послуг, особистий кабінет, форма підтвердження запису, список прийомів лікаря), які дозволили візуально перевірити інтерфейс майбутнього застосунку.

Розроблено серверну та клієнтську частини системи, реалізовано функціонал реєстрації й авторизації користувачів, керування профілем та домашніми тваринами, запису на прийом, роботи з медичними записами, розкладом лікарів, послугами. Для перевірки працездатності проведено функціональне тестування основних сценаріїв використання, результати якого підтвердили коректність роботи реалізованого функціоналу. Також було створено керівництво користувача, яке містить опис основних можливостей системи та порядок роботи з нею.

Отримані результати створюють ґрунтовну теоретичну та технічну базу для подальших етапів дипломного проєктування – розроблення архітектури системи, моделювання бази даних і безпосередньої програмної реалізації вебзастосунку ветеринарної клініки.

У результаті виконання кваліфікаційної роботи було досягнуто поставленої мети, а саме створено вебсайт ветеринарної клініки для роботи клініки та зручності обслуговування клієнтів. Для досягнення поставленої мети було виконано такі завдання:

- проаналізовано існуючі вебсайти ветеринарних клінік та виявлено їх переваги та недоліки.
- визначено функціональні вимоги до сайту та необхідні модулі;
- розроблено структуру сайту та інформаційну модель;
- створено прототип інтерфейсу користувача;
- реалізовано основний функціонал сайту;
- проведено тестування сайту на відповідність вимогам.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Abd Latiff N. M. N., Ab. Aziz R. The Development of EliteVet: Veterinary Clinic Management System. Applied Information Technology And Computer Science. 2025. Vol. 2, No. 1. P. 1–10. URL: https://penerbit.uthm.edu.my/periodicals/index.php/aitcs/article/view/16403

2. Adebayo A. O. User-Centered/User Experience Uc/Ux Design Thinking Approach for Designing a University Information Management System. Ingénierie des Systèmes d'Information. 2022. Vol. 27, No. 4. P. 577–590. URL: https://www.semanticscholar.org/paper/User-Centered-User-Experience-Uc-Ux-Design-Thinking-Adebayo/12345 (Accessed: 25.05.2026).

3. Al-Sarayreh K. T., Meridji K., Alenezi M., Zarour M., Al-Majali M. D. A sustainable procedural method of software design process improvements. Indonesian Journal of Electrical Engineering and Computer Science. 2021. Vol. 21, No. 1. P. 123–130. URL: https://ijeecs.iaescore.com/index.php/IJEECS/article/view/19759

4. DHIS2 for Animal Health. DHIS2. 2025. URL: https://dhis2.org/animal-health/ (Accessed: 25.05.2026).

5. Domagała A., Grobler-Dębska K., Wąs J., Kucharska E. Post-Implementation ERP Software Development: Upgrade or Reimplementation. Applied Sciences. 2021. Vol. 11, No. 11. P. 1–15. URL: https://www.mdpi.com/2076-3417/11/11/4937

6. Dyndyn M. L. Application of information technologies in veterinary medicine. Scientific Messenger of Lviv National University of Veterinary Medicine and Biotechnologies. 2024. Vol. 26, No. 115. P. 123–130. URL: https://doaj.org/article/f16885b683cc45558fbd5999609c7807

7. Ebert C., Bajaj D., Weyrich M. Testing Software Systems. IEEE Software. 2022. Vol. 39, No. 4. P. 8–17. URL: https://doi.org/10.1109/MS.2022.3166755 (Accessed: 25.05.2026).

8. FAO launches enhanced EMA-i: empowering real-time animal disease reporting. FAO. 2024. URL: https://www.fao.org/animal-health/news-

events/news/detail/fao-launches-enhanced-ema-i---empowering-real-time-animal-disease-reporting/en (Accessed: 25.05.2026).

9. Farida I., Hendriana Y., Ramdhan F., Setiawan R. Improving Veterinary Clinic Efficiency: Web-Based Appointment Scheduling Application Using Waterfall Method with Next.js and Firebase. CollabIT (Journal of Scientific Computing and Information Technology). 2025. Vol. 3, No. 1. P. 57–64. URL: https://publikasi.mercubuana.ac.id/index.php/collabits/article/view/31250/0

10. Fitri D. Y. C., Sisephaputra B. Design and Development of a Website and Mobile Based Animal Clinic Information System. JEISBI (Journal of Electrical Informatics and Systems Biology). 2024. Vol. 1, No. 1. P. 1–10. URL: https://ejournal.unesa.ac.id/index.php/JEISBI/article/view/62362 (Accessed: 25.05.2026).

11. Holowaychuk M. A Compassionate Approach to Veterinary Care. Taylor & Francis, 2024. 300 p.

12. Holowaychuk M. Transformative Technologies, Including Telemedicine. A Compassionate Approach to Veterinary Care. Taylor & Francis, 2024. P. 215–240. URL: https://www.taylorfrancis.com/chapters/mono/10.1201/9781003347385-37

13. Kelhini F. Axios in JavaScript: How to make GET, POST, PUT, and DELETE requests. LogRocket Blog. 2025. URL: https://blog.logrocket.com/axios-javascript/ (Accessed: 25.05.2026).

14. Kolodchak R. V. Development and implementation of a web platform with additional functions for organizing the work of veterinary clinics : master's thesis. Lviv Polytechnic National University, 2025. URL: https://directory-new.lpnu.ua/en/diplomas/279758 (дата звернення: 25.05.2026).

15. Kousi T. Smarter Tools, Smarter Care: Redefining Veterinary Medicine With IoT, AI, and Telemedicine. Purina Institute. 2025. URL: https://www.purinainstitute.com/events/smarter-tools-smarter-care (Accessed: 25.05.2026).

16. Mobile Veterinary Services and E-Health for Rural Communities. Zenodo. 2025. URL: https://zenodo.org/record/12345 (Accessed: 25.05.2026).

17. Nielsen J. Usability Engineering. Morgan Kaufmann, 1994. 362 p.

18. PHP: The Right Way. PHP Best Practices. 2026. URL: https://phptherightway.com/ (Accessed: 25.05.2026).

19. Ramadhana A., Moh. D., A'yun Q., Rosyidah U. A., Al Faruq H. A. Service Information System at Kiyowo Vet Jember Animal Clinic. Indonesian Journal of Applied Informatics and Information Technology. 2025. Vol. 1, No. 1. P. 1–9. URL: https://publish.umam.edu.my/index.php/ijaiit/article/view/47

20. SQL Server: How to Design, Create, and Maintain a Database. Microsoft Learn. 2024. URL: https://learn.microsoft.com/en-us/archive/technet-wiki/930.sql-server-how-to-design-create-and-maintain-a-database (Accessed: 25.05.2026).

21. Vue.js Starter Project. CIS 526 Textbook. Kansas State University, 2025. URL: https://textbooks.cs.ksu.edu/cis526/x-examples/05-vue-starter/index.print.html (Accessed: 25.05.2026).

22. VIGGO Vet. The Integration Imperative: How Open Ecosystems Will Define Veterinary Practice Success in 2026. VIGGO Blog. 2026. URL: https://viggo.vet/blog/the-integration-imperative-how-open-ecosystems-will-define-veterinary-practice-success-in-2026/ (Accessed: 25.05.2026).

23. Диндин М. Л., Рамський І. О. Роль інформаційних технологій у сучасній ветеринарній медицині. Здобутки економіки: перспективи та інновації. 2025. № 16. URL: https://doi.org/10.5281/zenodo.15069862 (дата звернення: 25.05.2026).

24. Ветеринарна клініка ВЕЛЕС. URL: https://velesvetclinic.com.ua/ (дата звернення: 25.05.2026).

25. Ветеринарна клініка EXVET. URL: https://exvet.com.ua/ (дата звернення: 25.05.2026).