

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці**

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему: **«РОЗРОБЛЕННЯ СИСТЕМИ ЕЛЕКТРОННОГО ЗАПИСУ ТА
УПРАВЛІННЯ ЧЕРГОЮ ДЛЯ ВЕТЕРИНАРНОЇ КЛІНІКИ»**

здобувачки Кищук Аліни Володимирівни

Науковий керівник:

д.е.н., професор

_____Мозгаллі О.П.

**Робота допущена до захисту перед
екзаменаційною комісією з атестації
здобувачів вищої освіти**

завідувач кафедри:

к.е.н., доцент

_____Тішков Б.О.

Київ 2025

Міністерство освіти і науки України
Київський національний економічний університет імені Вадима Гетьмана
Навчально-науковий інститут «Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»

галузь знань 12 «Інформаційні технології»

спеціальність 122 «Комп'ютерні науки»

ПОГОДЖЕНО:

Керівник проєктної групи(гарант)
освітньо-професійної програми

_____ Помазун О.М.

“ _____ ” _____ 2025 р.

ЗАТВЕРДЖУЮ:

Завідувач кафедри

_____ Тішков Б.О.

“ _____ ” _____ 2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувачки вищої освіти ***Кишук Аліни Володимирівни***

очної (денної) форми навчання

на підготовку кваліфікаційної бакалаврської роботи
на тему: «Розроблення системи електронного запису та управління чергою для ветеринарної клініки»

Тему затверджено наказом ректора Університету від « 7 » березня 2025 р.
№ 466- ст.

Кваліфікаційна бакалаврська робота виконується на матеріалах
отриманих під час навчання, в процесі проведення студентських досліджень й
шляхом аналізу наукових та літературних джерел з обраної тематики.

План кваліфікаційної бакалаврської роботи

Розділ I Характеристика та аналіз ветеринарної клініки.

Розділ II Розробка вимог і моделювання інформаційної системи.

Розділ III Проектування та реалізація компонентів системи.

Об'єкт дослідження. Процеси організації й автоматизації електронного запису, управління чергою та інформаційної взаємодії між персоналом і клієнтами у ветеринарній клініці.

Предмет дослідження. Методи, моделі та програмні засоби розроблення інформаційної системи електронного запису і управління чергою у ветеринарній клініці на основі сучасних вебтехнологій.

Мета кваліфікаційної бакалаврської роботи. Розробити інформаційну систему електронного запису та управління чергою у ветеринарній клініці для підвищення рівня обслуговування клієнтів, оптимізації навантаження робітників й надання легкої комунікації між адміністрацією й власниками домашніх тварин.

Конкретні завдання, які здобувач повинен виконати для досягнення поставленої мети:

У розділі I здобувач має дослідити предметну галузь діяльності ветеринарної клініки, виявити типові процеси, пов'язані з прийомом клієнтів, діями персоналу та обслуговуванням тварин. Необхідно проаналізувати джерела інформації, існуючі інформаційні системи у сфері ветеринарії та виявити недоліки або обмеження їхнього функціоналу. Слід встановити актуальні задачі, які потребують автоматизації, та обґрунтувати доцільність створення нової системи.

У розділі II передбачено формування вимог до інформаційної системи, які охоплюють функціональні, нефункціональні та бізнес-вимоги. Здобувач повинен виконати постановку задачі автоматизації процесів взаємодії між клієнтами та персоналом клініки, а також моделювання системи за допомогою нотацій UML та SysML. Необхідно побудувати діаграми варіантів використання, послідовності, класів, діяльності й розробити структуру бази даних і логічну архітектуру системи.

У розділі III необхідно реалізувати інформаційне, технічне та програмне забезпечення системи. Здобувач повинен розробити інтерфейсну частину на базі Blazor WebAssembly та серверну логіку на базі ASP.NET Core Web API, впровадити механізми реєстрації, авторизації, запису, перегляду черги тощо. Потрібно провести тестування функціоналу, оцінити практичність реалізованої системи та надати рекомендації щодо її впровадження у діяльність ветеринарної клініки.

Завдання підготував
науковий керівник
«_10_» березня 2025 р.

_____ **Мозгаллі Ольга Петрівна**

Завдання одержав
здобувач

_____ **Кищук Аліна Володимирівна**

«_10_» березня 2025 р..

Відгук
про кваліфікаційну бакалаврську роботу
здобувачки навчально-наукового інституту
«Інститут інформаційних технологій в економіці»
освітньо-професійної програми «Комп'ютерні науки»
Кищук Аліни Володимирівни

на тему
**«Розроблення системи електронного запису та управління чергою для
ветеринарної клініки»**

1. Актуальність теми: Тема дослідження є надзвичайно актуальною в умовах цифровізації ветеринарних послуг. Налагодження результативного керування потоком пацієнтів у ветеринарних клініках сприяє зменшенню черг, підвищенню рівня обслуговування та оптимізації роботи персоналу.
2. Позитивні риси кваліфікаційної бакалаврської роботи: Робота характеризується чіткою структурою, аргументованою постановкою проблеми, логічною побудовою рішень, використанням сучасних технологій веброзробки. Реалізована частина системи демонструє функціональну завершеність клієнтського сегмента.
3. Наявність самостійних розробок автора. Авторка Кищук А.В. самостійно виконав проєктування архітектури, бази даних, реалізацію клієнтського функціоналу, включаючи модулі реєстрації, авторизації, запису на прийом і перегляду історії записів.
4. Цінність теоретичних висновків та практичних рекомендацій: Теоретичні висновки ґрунтуються на аналізі сучасних стандартів проєктування систем і можуть бути використані для побудови подібних рішень у медичній сфері. Практична реалізація дозволяє апробувати систему на реальних прикладах, що підтверджує її результативність.
5. Наявність недоліків: До незначних недоліків розробленої системи можна віднести відсутність інтеграції з платіжними сервісами. У перспективі доцільно розширити систему для підвищення зручності користування й загальну привабливість системи.
6. Загальна оцінка кваліфікаційної бакалаврської роботи та її допущення до захисту перед ЕК: Робота оформлена згідно встановлених вимог, відповідає меті та завданням дослідження, демонструє самостійність студента та здатність реалізації сучасних рішень. Кваліфікаційна робота заслуговує на високу оцінку та допускається до захисту перед ЕК.

Науковий керівник

професор, д.е.н. Мозгаллі О.П.

“14” червня 2025 р.

Рецензія

на кваліфікаційну бакалаврську роботу
здобувача вищої освіти

Кищук Аліни Володимирівни

Тема: «Розроблення системи електронного запису та управління чергою для ветеринарної клініки»

Актуальність теми кваліфікаційної бакалаврської роботи і доцільність її розроблення. Тема бакалаврської роботи є актуальною, оскільки автоматизація процесів електронного запису і керування чергою в ветеринарних установах є важливою складовою цифрової трансформації галузі. На ринку України відчутна потреба у подібних рішеннях, які дозволяють оптимізувати обслуговування клієнтів та зменшити навантаження на персонал. Доцільність розроблення підтверджується зростанням попиту на сучасні вебсистеми у цій сфері.

Якість проведеного дослідження. Проведено глибоке дослідження предметної області і аналіз існуючих аналогів, застосовано системний та об'єктно-орієнтований підходи до проєктування, використано сучасні інженерні й програмні засоби. Побудовані діаграми, описані сценарії, часткова реалізація системи свідчить про практичне розуміння поставленої задачі.

Позитивні риси кваліфікаційної бакалаврської роботи. До позитивних рис слід віднести чітке формулювання вимог до системи, грамотне структурування матеріалу, коректне застосування стандартів моделювання. Також відзначається самостійність автора у формуванні архітектури системи та створенні діаграм. Зауваження. У майбутньому доцільно розширити систему, включивши керування чергами, адміністрування записів та інші сервіси, які забезпечать повноцінну функціональність.

Практична значимість висновків і рекомендацій. Отримані результати мають практичну значимість, оскільки можуть бути адаптовані для використання у ветеринарних клініках. Висновки щодо побудови архітектури, розмежування ролей користувачів та застосування сучасних вебтехнологій є корисними для впровадження у реальні проєкти. Робота оформлена згідно встановлених вимог і заслуговує на «відмінну» оцінку.

ВСП «Фаховий коледж інформаційних систем і технологій КНЕУ імені Вадима Гетьмана»

Завідувач відділенням

техніко-економічних спеціальностей _____ Правдивий А.М.

АНОТАЦІЯ

кваліфікаційної бакалаврської роботи
здобувачки першого (бакалаврського) рівня вищої освіти, 4 курсу,
виконаної на тему: «Розроблення системи електронного запису та управління
чергою для ветеринарної клініки»
Київ: кафедра інформаційних систем в економіці, 2025 р.

Кваліфікаційна бакалаврська робота присвячена розробленню інформаційної системи електронного запису та управління чергою для ветеринарної клініки, що є актуальним завданням у зв'язку з впровадженням цифрових рішень у медичну сферу.

Робота складається з трьох логічно пов'язаних розділів.

У першому розділі охарактеризовано контекст і об'єкт вивчення функціонування ветеринарної клініки, виконано аналіз сучасного стану автоматизованих систем у сфері ветеринарії, виявлено проблеми та переваги.

У другому розділі проаналізовано і сформовано вимоги, здійснено постановку задачі та побудовано моделі системи за допомогою UML та SysML.

Третій розділ містить опис реалізації компонентів: інформаційного, технічного та програмного забезпечення системи. Розроблення здійснено з використанням ASP.NET Core Web API, Entity Framework, Blazor WebAssembly та Microsoft SQL Server.

У висновках наведено результати та рекомендації щодо впровадження системи.

РЕФЕРАТ

Кваліфікаційна бакалаврська робота містить 91 сторінку, 15 таблиць, 57 рисунків, список використаних джерел з 32 найменувань, 3 додатки.

«Розроблення системи електронного запису та управління чергою для ветеринарної клініки»

Об'єкт дослідження: процеси організації й автоматизації електронного запису, управління чергою та інформаційної взаємодії між персоналом і клієнтами у ветеринарній клініці.

Предмет дослідження: методи, моделі та програмні засоби розроблення інформаційної системи електронного запису і управління чергою у ветеринарній клініці на основі сучасних вебтехнологій.

Мета і завдання дослідження: розробити інформаційну систему електронного запису та управління чергою у ветеринарній клініці для підвищення рівня обслуговування клієнтів, оптимізації навантаження робітників й надання легкої комунікації між адміністрацією й власниками домашніх тварин.

Відповідно за метою окреслено такі завдання:

- Провести аналіз предметної галузі та вивчити сучасний стан автоматизації ветеринарних клінік.
- Визначити основні вимоги до функціональності системи та сформулювати завдання автоматизації.
- Створити концептуальну модель системи, діаграми варіантів використання, послідовностей, класів, діяльності.
- Розробити архітектуру системи, структуру бази даних та реалізувати її компоненти за допомогою технологій ASP.NET Core Web API, Blazor WebAssembly і SQL Server.
- Надати реєстрацію та авторизацію користувачів, функціонал запису, перегляду черг, зміни черговості та адміністрування.
- Провести тестування системи та надати рекомендації щодо її впровадження у ветеринарній практиці.

Теоретична, методична та практична значущість отриманих результатів.

Теоретичне значення роботи в її узагальнених підходах до моделювання інформаційних систем для ветеринарних клінік, а ще у розробці моделі автоматизованої системи запису та управління чергою.

Методична цінність у застосуванні стандартів побудови вимог SysML, UML, задля цілісного та структурованого підходу розроблення.

Практична значущість визначається, як реалізована система може бути використана в реальних ветеринарних клініках оптимізувавши взаємодію персоналу з клієнтами, зменшивши час очікування та комфорт при плануванні відвідувань.

Рік виконання кваліфікаційної бакалаврської роботи – 2025.

Рік захисту роботи – 2025.

Ключові слова: ветеринарна клініка, електронний запис, управління чергою, інформаційна система, автоматизація, SQL Server, ASP.NET Core Web API.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	4
ВСТУП.....	5
РОЗДІЛ 1	8
ХАРАКТЕРИСТИКА ТА АНАЛІЗ ВЕТЕРИНАРНОЇ КЛІНІКИ.....	8
1.1 Контекст і об'єкт вивчення функціонування ветеринарної клініки.....	8
1.2 Аналіз літературн та практики використання ІС і технологій у ветеринарній галузі.....	14
РОЗДІЛ 2	21
РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	21
2.1 Аналіз і специфікація вимог до інформаційної системи	21
2.2 Постановка та алгоритм розв'язання задачі.....	24
2.2.1 Постановка задачі	24
2.3 Моделювання інформаційної системи.....	34
2.3.1 Моделювання поведінки системи	34
2.3.2 Моделювання структури системи	46
2.3.3 Розподіл вимог за компонентами системи	50
РОЗДІЛ 3	54
ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ	54
3.1 Інформаційне забезпечення	54
3.1.1 Загальна характеристика інформаційного забезпечення.....	54
3.1.2 Організація збору і передання первинної інформації	57
3.1.3 Побудова системи класифікації та кодування	58
3.1.4 Проєктування форм первинних документів та відеокадрів	59
3.1.6 Вибір СКБД	67
3.1.7 Інфологічна модель бази даних	67
3.2 Технічне забезпечення.....	71
3.2.1 Загальні положення та схема автоматизації.....	71
3.2.2 Структура комплексу технічних засобів	72
3.2.3 Опис автоматизованого робочого місця.....	74
3.2.4 Схема мережі передачі даних	75

3.3 Програмне забезпечення	77
3.3.1 Структура програмного забезпечення	77
3.3.2 Системне програмне забезпечення	78
3.3.3 Прикладне програмне забезпечення	80
3.3.4 Програмна документація	81
3.4 Результати реалізації інформаційної системи.....	83
ВИСНОВКИ.....	87
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	90

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ІЗ – інформаційне забезпечення

ІС – інформаційна система

ПЗ – програмне забезпечення

ТЗ – технічне забезпечення

ОС – операційна система

ВСТУП

Актуальність теми. Наразі у всьому світі зростає частка використання інформаційних впроваджень. Дуже стрімко з'являється потреба в автоматизації адміністративних процесів у різних сферах життя, у сфері охорони здоров'я, як от у ветеринарній медицині. Необхідним стає своєчасний комфортний доступ до ветеринарних послуг з оптимізованою організацією роботи клінік, важливі критерії якості обслуговування власників улюбленців. Не дивлячись на це, велика кількість українських ветеринарних закладів досі чомусь, немає інноваційних систем електронного запису та управління чергою. Ручне виконання роботи чи старе ПЗ призводить до проблем, слабкої віддачі, плутанини в графіках і потім, втрати клієнтів. Взявши до уваги державний напрямок розвитку цифрових інструментів, розроблення вебзастосунку автоматизації таких процесів є розумним і критично необхідним планом довготривалого розвитку ветеринарного обслуговування в Україні.

Аналіз останніх досліджень і публікацій. Науково-технічна спільнота активно досліджує питання цифрової трансформації медичних та ветеринарних установ. У працях часто пишуть про: проєктування медичних інформаційних систем, інтеграції баз даних пацієнтів чи інших систем, захист інформації, будову архітектури клієнт-серверної програми. У статтях наукових розбирають використання REST API і інтерфейсів в системах запису, адже специфіка має мати свій принцип. Зараз існує проблема у недостатності рішень, що не ігнорують усі етапи взаємодії, гнучкого управління чергою з підтримкою сценаріїв обслуговування.

Мета і завдання дослідження. Метою кваліфікаційної бакалаврської роботи є розробити інформаційну систему електронного запису та управління чергою у ветеринарній клініці для підвищення рівня обслуговування клієнтів, оптимізації навантаження робітників й надання легкої комунікації між адміністрацією й

власниками домашніх тварин. Система має бути орієнтована на потреби середніх ветеринарних клінік в Україні.

Об'єкт дослідження. Процеси організації й автоматизації електронного запису, управління чергою та інформаційної взаємодії між персоналом і клієнтами у ветеринарній клініці.

Предмет дослідження. Методи, моделі та програмні засоби розроблення інформаційної системи електронного запису і управління чергою у ветеринарній клініці на основі сучасних вебтехнологій.

Методи дослідження. Задіяно методи системного аналізу, моделювання систем UML й SysML, методи об'єктно-орієнтованого проектування, інструменти інженерії. Реалізація ґрунтується передовими вебтехнологіями ASP.NET Core Web API, Blazor WebAssembly, SQL Server, є методи порівнянь аналізу рішень і метод оцінювання функціональності з сценаріями використання.

Теоретична, методична та практична значущість отриманих результатів. Теоретичне значення роботи в її узагальнених підходах до моделювання інформаційних систем для ветеринарних клінік, а ще у розробці моделі автоматизованої системи запису та управління чергою. Методична цінність у застосуванні стандартів побудови вимог SysML, UML, задля цілісного та структурованого підходу розроблення. Практична значущість визначається, як реалізована система може бути використана в реальних ветеринарних клініках оптимізувавши взаємодію персоналу з клієнтами, зменшивши час очікування та комфорт при плануванні відвідувань.

Інформаційна база роботи включає результати аналізу діяльності ветеринарних клінік, публікації науковців, документації з розроблень вебдодатків, законодавчих актів, стандартів, більше того використано дані офіційних вебсайтів ветеринарних закладів.

Апробація результатів роботи. Ідея дослідження була представлена на студентській науково-практичній конференції факультету інформаційних

технологій. Робота опублікована. У роботі розглядались технології розроблення системи ветеринарні клініки різноманітними технологіями. Переглянути тези можна в Додатку А.

Структура роботи. Загальний обсяг роботи становить 91 сторінки друкованого тексту, 57 рисунків на 32 сторінках, 15 таблиць на 10 сторінках, 3 додатки на 32 сторінках. Список використаних джерел налічує 32 найменування. Робота має вступ, три розділи, висновки, список джерел, додатки.

РОЗДІЛ 1

ХАРАКТЕРИСТИКА ТА АНАЛІЗ ВЕТЕРИНАРНОЇ КЛІНІКИ

1.1 Контекст і об'єкт вивчення функціонування ветеринарної клініки

Перш за все дослідження починається для всіх систем однаково. Першим етапом варто дізнатися функції і роботу галузі для якої будується інформаційна система. За темою бакалаврської роботи предметною галузю є ветеринарна клініка і процеси надання ветеринарних послуг.

Ветеринарна клініка - заклад для медичного обслуговування та лікування тварин[1].

Сфера різноманітна до неї входять не лише діагностики, профілактики і процеси лікування захворювань, важливим моментом можна виокремити організаційне забезпечення обліку, управління записами власників, оброблення інформації пацієнтів.

Ветеринарія займається профілактикою здоров'я, діагностуванням та лікуванням захворювань або травм тварин[2].

Ветеринарна медицина допомагає уникати поширення хвороб, адже вони можуть передаватися людині, ще вона контролює якість продуктів тваринного походження, має тісні зв'язки з іншими галузями, як от сільське господарство.

У ветеринарній галузі розвиток технологій робить внесок в автоматизацію центральних процесів, скажімо як: електронне ведення історій хвороби, онлайн запис на прийом, електронні рецепти і навіть, контроль залишків медикаментів, тобто сприяють підвищенню точності обслуговування, зменшують можливість виникнення помилки від процесів, якими займається лише людина.

Раніше вже впроваджені в ветеринарній сфері інформаційні системи відкривають шлях до збирання статистичних для моніторингу дієвості лікування даних та аналізу ситуації в країні. Системи ще застосовують для можливості мати віддалений доступ до даних, важливо у випадках мобільних клінік або наданні дистанційні медичних консультацій[3]. Предметна галузь охоплює клінічні, інформаційно-аналітичні компоненти, вони ж потребують глибокого дослідження і технологічного поліпшення.

Переходимо до об'єкта дослідження. Їм виступає саме автоматична реалізація процесів у ветеринарній клініці, взаємодія між адміністративним складом, ветеринарами, асистентами та власниками улюбленців. В процесі виконання послуг лікування в клініці, наявні деякі основні бізнес-процеси: реєстрація нових клієнтів і їх тварин, запис на прийом з вибором різних параметрів, формування черги, створення з оновленням і зберіганням медичних даних про улюбленців, проведення діагностики, лікування, обліку медикаментів і т.д. Дослідження великих систем може включати процеси управління внутрішніми ресурсами клініки, формування графіків роботи ветеринарів, комунікацію підрозділів.

Догляд за домашніми тваринами так само важливий для ветеринарної галузі, як і догляд за людьми[4].

Метою в цій сфері є організація безвідмовної роботи ветеринарного закладу шляхом автоматизації інформаційної системи. Системи зазвичай мають формальні елементи електронні бази даних, шаблони карток улюбленців, інтерфейси користувачів та зв'язки між тим, як все влаштовано між елементами системи. Окреме значення посідає механізм прийняття рішень на всіх рівнях. Він починається від розроблень управлінських планів до медичних приписів, які потребують нагальної інформації з підтримкою від ІС. Аби реалізувати таку систему слід врахувати винятковість клінічної практики, правила збереження

таємниці інформації, оперативність доступу інформації з можливістю розширення коли потрібно.

Враховуючи, що ветеринарна клініка має велику кількість різних процесів, дослідження зосереджено на розробленні можливості електронного запису та управління чергою. Система повинна бути зрозумілою та підтримувати весь етап обслуговування.

Предметна область включає в себе повністю як внутрішні структури ветеринарних клінік, як от організаційна структура, розподіл ролей, внутрішня документація[5].

Також містить і зовнішнє середовище з правовими нормами, конкуренцією середовищ, сектором препаратів. У сфері ветеринарії працюють різні клініки державні під органами влади, приватні з усім діапазоном послуг для власників улюбленців. Досить часто ветеринарні клініки працюють з лабораторіями, аптеками, зоомагазинами, розширюються в локальній середовищі обслуговування.

Настільки всеохопний зв'язок спонукає до єдиного управління накопичень і передачі відомостей, створенню гармонійної взаємодії різних між собою систем і інтерфейсів.

У напрямку зоомедичної галузі еволюція дистанційних послуг медицини виводить на перший план необхідність в новітніх рішеннях, що зможуть надати можливість проводити консультації й зберігати необхідну документацію в онлайн. Треба відмитити те, що ветеринарна система потребує врахування сезонних навантажень, звернень навесні або восени.

Навчання персоналу в роботі із новими платформами, теж варто відмитити, адже комфорт користування системою залежить від її продуктивності. Інший момент - взаємодія з клієнтом. Вона починається від першого звернення до отримання результатів лікувань в ветеринарних клініках і має бути безпечною. Тому, слід розглядати предметну область не лише з технічної, а разом з стороною користувацького досвіду. Група зазначених характеристик робить сферу

ветеринарних послуг складною, але разом з тим перспективною в центрі діджиталізації.

Ухвалення рішень - процес вироблення рішень, розумова діяльність із визначення результату такої дії[6].

Середовище прийняття рішень у ветеринарній клініці охоплює концептуальне планування з оперативним реагуванням на певні клінічні ситуації. Ухвалюються рішення на протязі обмеженого часу, неповної інформації, різних рівнів ризиків. Ветеринар, тим часом, обирає можливі варіанти при лікуванні, приділивши увагу попередній історії хвороби улюбленця, чи є алергія, які послуги зможе оплатити власник тварини та наявність препаратів на складі. Адміністратор приймає рішення змінити порядок запису, обслуговувати пацієнта в критичному стані поза чергою або перенести візит до іншої черги. Курс розвитку формує керівництво займається плануванням закупівель, аналізує роботу персоналу. Кожне рішення базується на доступності даних, через це і якість, і повнота з актуальністю має критичне значення для інформації, крім того для цього потрібна інтегрована ІС з підтримкою різних типів рішень: стандартних, критичних, ризикованих та сумнівно структурованих[7].

Ветеринарна практика сьогодні бореться з швидкозмінним середовищем[8]. Система має швидко надати доступ до всієї історії пацієнта з попередніми діагнозами підчас критичних ситуацій. Якщо говорити про можливу інтеграцію з лабораторіями або зовнішніми сервісами, треба їй передбачити відповідні канали обміну даними з ними. Відповідно тоді, можна сказати, що середовище прийняття рішень є основою результативного функціонування клініки. Воно має бути враховано при проєктуванні.

Фактори впливу на прийняття рішень. Процес завжди відбувається в певному просторі під впливом численних факторів, політичних, економічних, правових умов тощо. Економічна нестабільність може мати необхідність в постійнім коригуванні рішень, проте політична невизначеність або правові обмеження

можуть зменшити можливі альтернативності. Інформаційний технологічний розвиток здатен розширити діапазон наявних рішень, через автоматизацію або аналізу даних, або при моделюванні сценаріїв. Очікування від громадян та культурні особливості, до того ж, теж мають істотний вплив в місцях, що стосуються людських потреб. Характер впливу всіх елементів може бути прямим, опосередкованим, сильним або навпаки слабким, стабільним чи змінним. Потрібно не прогавити момент і вчасно знайти ці фактори, визначити їх вплив й враховувати їх динаміку.

На практиці рішення дуже рідко ізольовані в більшості випадків вони поєднані і будують логічну послідовність. Прийняті раніше вже ж встановлюють обмеження для майбутніх дій, можуть мати вплив із затримкою. Ось при обранні постачальника вони змінюватимуть ґатунок продукції, терміни її поставки з витратами. Коли поточна ситуація є наслідком попередньої, слід враховувати залежність рішень і спиратися на "залежність шляху". Поняття "залежність від шляху", є дуже загальним і стосується як наслідків розвитку, так і соціальної динаміки[9]. Тут же значення має і вплив паралельних або супутніх задач. Також, буде дивно не врахувати значущі впливи рішень прийнятих в сьогоднішній день. Вони можуть закласти початок можливостей або загроз у перспективі. Все це буде складну систему, в якій кожне рішення змінює умови для наступного і тому, варто мати системний підхід планування ухвалення рішень, з даними про лаги часу й зворотні зв'язки. Оцінка майбутніх наслідків при плануванні впливу рішень має включити короткострокові наслідки з довгостроковим планом аналізу.

Прийняття обґрунтовано кращих рішень неможливе без всіх інформаційних матеріалів. Інформація буває внутрішня й зовнішня. Перша надходить зсередини будь-якої організації, остання з оточення[10]. Джерела звідки надходять дані охоплюють звіти, бази даних, соцмережі і загальнодоступні дані статистики. Застосовуються різні підходи, щоб мати в результаті релевантну інформацію.

Ще варто відзначити важливість типів в даних. Емпіричні зазвичай отримуються на практиці, або з власного досвіду. Об'єктивні є фундаментом досліджень, аналізу та прийняття обґрунтованих рішень.

Формальні процедури, евристика, імітаційне моделювання через всіх них, може проводитися оброблення для даних. Всі мають свої переваги та обмеження, і можна зустріти їх комбінацію - комбінований підхід. Повинні існувати тільки якісні дані, щоб уникати упередженості й неповноти, саме така база зменшує невизначеність і нарощує точність для рішень.

Ситуації можуть мати різну складність та контекст, тому і підходи до їх аналізу, і розв'язання під час прийняття рішень можуть бути різними. Закриті задачі мають відомий кожен параметр, обмеження та варіанти дій, тому вони вирішуються точними методами. Тим часом у відкритих, частка змінних невідома, а цілі можуть змінюватися.

Типи рішень ще залежать від міри визначеності[11]. Детерміновані задачі мають прогнозовані наслідки, рішення за умов ризику можуть мати невідомі підсумками. Наявність взагалі повністю нечітких розмитих цілей утворює труднощі в генерації критеріїв, тому повинно мати адаптивні стратегії. Посилені відкриті рішення взагалі розраховано не тільки на вибір, а й породженні оновленої будови проблеми. Відмінною ознакою середовища сучасності виступає саме зростання кількості рішень при невизначеності, яка потребує від суб'єктів нових здібностей. Не існує ідеального підходу для таких ситуацій, кожна з усіх можливих ситуацій може вимагати індивідуальних методів.

1.2 Аналіз літератури та практики використання ІС і технологій у ветеринарній галузі

У охороні здоров'я застосування інформаційних систем у ветеринарну практику стало особливо доречно. ІС тут зорієнтовані на автоматизацію процесів обліку, управління чергами, контролю і аналітичного оброблення клінічних матеріалів. Наукова література викладає, як зростає значення діджитал платформ в підвищенні стандарту послуг й зниженні витрат на опрацювання даних.

Посеред напрацювань у галузі ветеринарної медицини застосовуються найбільше прогресивні розроблення як-от комп'ютерні системи: Hippo Manager, eVetPractice, NaVetor. Можна сказати, що всі мають свої обмежувальні рамки й сильні сторони.

Hippo Manager містить інтуїтивний інтерфейс, хмарне зберігання, можливості інтеграції різних модулів, але попри все це її набір функцій дуже малий для малих клінік з маленькими бюджетними коштами.

Платформа для ветеринарних клінік eVetPractice націлена на інші моменти. Вона сфокусована на адаптивності настройки всіх робочих процесів.

Належним чином має гарний вплив для широко профільних закладів. Із мінусів, використовує набагато більше часу на навчання працівників для користування нею.

Останньою розглянемо NaVetor. Система дуже комфортна у використанні через свої модулі запису на прийом і наявності в собі звичайні вихідні опції CRM, управління взаємовідносинами з власниками улюбленців, хоча і має проблеми з наявністю персоналізації[12].

Так цього бракує, але і без цього вона досить дієва передова система.

В Україні, науковці акцентують звернути увагу на адаптацію ІС до внутрішніх умов. Підкреслити особливість мовної підтримки в країні,

документообігу й правових вимог. Закордоном більше приділяють увагу на аналіз типізації даних, безпеці оброблення медичних відомостей і застосуванні аналітики. Відсутній єдиний підхід до інтеграції ІС, тому і залишаються проблема злиття з іншими елементами ІТ в ветеринарних лікарях, такими як, наприклад, є лабораторії або ж аптеки.

У всесвітній мережі, Інтернеті, існує велика кількість різноманітних статей з ветеринарної інформатики, там відзначають недостатній рівень взаємодії між ПЗ системи із користувачами. Є підстави вважати, що це є наслідок, який знижує результати автоматизації.

Розроблення оптимальної ІС потребує попереднього аналізу функціональних вимог системи. Проаналізувавши літературні джерела, доречно зазначити, що треба врахувати під час розроблення не лише організацію певних процесів, а й уможливити отримання в кінці роботи позитивний відгук користувачів. Поточним є питання про рівень компетентності робітників у розумінні і володінні інформаційними технологіями, чи можуть клієнти вільно взаємодіяти з системою, чи існує наявність технічної інфраструктури чи ні.

При критичному розгляді наукової літератури знайшлися основні концепти та підходи розроблення ІС у сфері надання ветеринарної медицини. Електронна реєстрація улюбленців, управління чергами записів, оцінка дієвості успішності результативності терапевтичних програм, оцифрування руху потоку документів, саме ті кілька основних розділів аналізу на які вказують завжди експерти. Більшість робіт присвячено безпосередньо недостатній інтегрованості, обмеженості в наборі функцій для цілковитого покриття потреб в клініці. Практика у використанні ІС сьогодні показує, які основні проблеми виникають у багатьох ветеринарних закладах. Вони мають справу з конфліктами сумісності модуль з модулем, при незрозумілій і заплутаній настройки, і ще з високою вартістю реалізації системи в життя.

Із наявних позитивних сторін є ті, при яких, розроблення платформ для ветеринарної галузі підвищило значно оперативність обслуговування, покращило комунікацію між користувачами системи, зменшило в декілька разів обсяг роботи адміністрації.

Ще в літературі можливо знайти модель економічного прогнозування для розрахунку потрібного числа ресурсів чи алгоритми, які планують завантаженість клінік.

Ряд досліджень з усіма подробицями розбирають тему створень місць АРМ із урахуванням ролей співробітників. Виділяють вхідні/вихідні потоки даних: форми реєстрації, запити, звіти, медкартки[13].

Аргументований підхід розроблення системи спирається на 2 елемента теорію і практику з реального досвіду. Головним є комплексний підхід, адже він містить в собі всі етапи від запису клієнта на прийом ветеринара до отримання ним медичних документів. Тут з метою підвищити виваженість керівницьких рішень, практикують економіко-математичні моделі, наприклад, моделі оптимізації графіку прийому або прогнозування завантаженості, з ними також, методи лінійного програмування, кластеризації і статистичних методів[14]. Раціональність рішень переважно є логікою оброблення та отриманими результатами.

Перейдемо до інформаційного забезпечення.

Спочатку варто розібратися, де вхідні та де вихідні потоки такі як: заявки на якійсь запит від будь-кого з користувачів системи, реєстраційні форми, форми запису, медичні картки, і т.д. Структура всіх них має відповідати вимогам до уніфікації і поліпшити точність введених вручну даних з запитами. Між тим, ПЗ системи має бути можливим до масштабування, мати легко зрозумілий інтерфейс, відповідати стандартам безпеки.

Технічне забезпечення системи поширюється на сервери, мережеве обладнання і периферійні пристрої АРМ або клієнта[15].

Елементи ТЗ повинні узгоджуватись до єдиної архітектури системи, мати розподіл функцій між автоматизованими робочими місцями на 2 зони: реєстратура й адміністрація, при розробленні системи для великої кількості можлива третя зона мед кабінети.

Місцем, де виникає інформація є перший контакт з користувачем. Періодичність обробки інформації має відношення до типу задачі. При виконанні однієї дані можуть фіксуватися в реальному часі, в інших щодня або взагалі раз в кожен місяць. Користувачами ІС виступають звичайно троє: адміністратори, ветеринари і клієнти.

Детально порівняємо і розглянемо існуючі системи віднесені до ветеринарних систем або платформ. Проаналізуємо як працює низка зарубіжних систем, що мають поточні перспективи в інформатизації.

Наразі популярними системами для ветеринарної сфери є такі, як Hippo Manager, NaVetor й eVetPractice. Всі вони масово працюють ІС для ветеринарних клінік та підтримують автоматизовані клінічні і не тільки процеси для закладів.

Після того як було розглянуто раніше наукову з прикладною літературою, стало зрозуміло, що варто визначитись з сукупністю таких підходів до будови ІС, серед яких переважають сервісно-орієнтована архітектура, хмарні технології і мобільність. При ґрунтовному аналізі демонстровано як, багато існуючих платформ мають схожі функції з різною можливістю інтеграції і ступенем адаптації під потреби.

Наприклад, якщо повернутися до українських систем з ветеринарними послугами, тут дійсно існує потреба в повнофункціональних системах, які могли б інтегрувати запис, управління чергою, базу, модуль CRM з модулем аналітики всі разом в єдиному рішенні. Можна зробити висновок того, як є потреба у розробці місцевої, спеціально розробленої під ринок ветеринарних послуг України системи з найкращими практиками досвіду інших країн.

Вирішено почати аналіз існуючих інформаційних систем з Hippo Manager.

Як відомо, систему знають дуже гарно у інформаційному просторі, її називають прикладом зразкового поєднання легкого інтерфейсу, функцій електронного медичного запису, наявності обліку медикаментів і має все потрібне для нагляду за фінансовою діяльністю.

Є можливість дізнатись відгуки користувачів, які вже користувались послугами системи[16]. Власникам улюбленців Hippo Manager подобається, вони відзначають її комфорт при користуванні, мінімум потреби в навчанні персоналу та відмінну пристосованість до телефонів. Вона може керувати всіма прийомами в клініці, додаючи ресурси, змінювати розклад зустрічей і багато іншого[17].

Платформа має модуль планування прийомів, він може вести чергу, контролювати доступність ветеринарів й автоматизувати сповіщення для власників улюбленців. Під час знаходження статей про ІС, з'явилась важлива інформація про деякі знайдені обмеження до неї. Найгучнішими є обмеження в широкому спектрі налаштувань документів системи та скупа кастомізація серед інших аналогічних проєктів. Навіть зважаючи на те, що Hippo Manager дійсно має проблеми, все одно система масово поширена через свою модель, яку можна ж адаптувати і для роботи в Україні. Застосовує концепцію SaaS, має постійну можливість працювати, тобто гарний варіант для вибору та прийняття рішень у динамічному середовищі ветеринарного бізнесу.

eVetPractice, друга система обрана для розгляду в роботі.

eVetPractice - це комплексне рішення для управління ветеринарною практикою[18]. Система відзначена високою мірою автоматизації адміністративних і клінічних процесів. Вона вирізняється своєю архітектурою серед інших, модульною архітектурою. Інша відмінність у великим набором різноманітних аналітичних інструментів, які полегшують роботу для робітників.

Найголовніший плюс eVetPractice, наявність опції інтегрування з лабораторіями, корисно відображати аналізи потім в кабінетах власників для їх тварин. Взагалі система містить майже все аби назвати себе повною. В evetpractice

процеси починаються від первинної реєстрації до видачі рецепту з контролем одужання. eVetPractice надає зручний інтерфейс для оптимізації процесу запису на прийом[19]. Система має свою, так скажімо родзинку, вона в собі має механізм прийняття рішень який працює на алгоритмах, які займаються врахуванням історії хвороб перед лікуванням.

Після аналізу наявності аналітичного ядра та розгалуженої бази знань для eVetPractice робить її могутнім засобом при роботі з ветеринарними клініками. Єдиний мінус її в потребі численних ресурсів для роботи, через це її менш любляють на українському ринку.

Система NaVetor. Посеред списку систем вона наймолодша в порівнянні з eVetPractice й Hippo Manager, навіть якщо порівнювати з всіма виступає новим креативним варіантом. NaVetor орієнтується своєю простотою та легкою для сприйняття логікою роботи з інтерфейсом користувачам.

Вона розроблена, щоб допомогти вести медичну документацію та легко керувати своїм бізнесом[20]. Будова проєкту містить відомі, а головне релевантні стандарти UX/UI-дизайну.

Із недоліків, а правильніше сказати обмежень, знайдено в інтеграції з базами і з надто великою кількістю мов.

Використовують NaVetor в умовах невизначеності. Система дає змогу стрімко змінити процес прийому при потребі, реагує на динаміку в зміні черги записів. Розроблений для простоти використання, забезпечуючи швидкий доступ до основних функцій[21].

Перевагою системи NaVetor є пристосування до довільної будови процесів до конкретних клінік. Другий момент в наявному сполученні з бухгалтерськими чи лабораторними елементами, сторонніми системами. Вага системи в її успішній роботі, змозі змінювати певні процеси.

NaVetor зазвичай в різних документах пишеться системою, яка фокусується на гарну картинку, використовує індивідуалізовані панелі і має функціонал модифікації карток улюбленців.

При аналізі матеріалів пов'язаних з ветеринарною клінікою і спостережень у даній сфері, можна сформулювати головну задачу дослідження розроблення інформаційної системи електронного запису та управління чергою для ветеринарної клініки. Метою є створення системи з продуктивним організуванням роботи клініки, прямою та чіткою взаємодією з клієнтами і високу якість в створенні і управлінні записами. Актуальність роботи полягає життєвій потребі до діджиталізації ветеринарної галузі, бо вона поки сильно відстає від темпів інформатизації у медицині.

РОЗДІЛ 2

РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Аналіз і специфікація вимог до інформаційної системи

Формулювання вимог, яке перекладає або виражає потребу та пов'язані з нею обмеження та умови[22]. Вони визначають функції роботи системи, проблеми що має вирішувати і які очікування замовника та кінцевих користувачів має задовольняти. ІС електронного запису та управління чергою для ветеринарної клініки повинна мати практичне та дієве управління записами клієнтів, автоматизувати їх обробку, зменшити навантаження на адміністраторів клініки та зробити сервіс сучаснішим і привабливішим для власників улюбленців. Вимоги системи структуровано відповідно до стандарту SysML ISO IEC 19514:2017 використавши діаграми й засоби Enterprise Architect.

Sparx Systems Enterprise Architect - це надійний, багатофункціональний, гнучкий і комплексний інструмент моделювання для всього процесу розробки[23].

Основні категорії вимог поділено на три блоки: бізнес-вимоги, функціональні, нефункціональні.

Бізнес-вимоги визначають стратегічну мету створення системи[24]. Головною метою є покращення якості обслуговування клієнтів та збільшення продуктивності внутрішніх процесів ветеринарної клініки. До бізнес-вимог входять:

- Збільшення кількості нових клієнтів. Система повинна мати зручний і потрібний функціонал, що сприятиме залученню нових клієнтів.

- Зниження адміністративних витрат. Автоматизація процесів запису, ведення черг і перегляду історії відвідувань має скоротити обсяг адміністративної роботи.
- Підвищення продуктивності персоналу. Система має зменшити навантаження на адміністраторів шляхом оптимізації черг.
- Збільшення кількості повторних візитів. Надання клієнтам можливості переглядати історію записів й здійснювати нові, має стимулювати їх повторно звернутися до ветеринарної клініки в разі потреби.
- Підвищення доходу клініки. Реалізація системи електронного запису повинна позитивно вплинути на загальний дохід клініки.

Функціональні вимоги описують, що саме повинна робити система її поведінку у відповідь на дії користувачів. До функціональних входять вимоги:

- Реєстрація клієнта. Система повинна мати можливість створення облікового запису нового клієнта.
- Авторизація користувачів. Реалізована система автентифікації має дозволяти вхід до системи як клієнтам, так і адміністраторам.
- Оформлення запису на прийом. Користувач повинен мати змогу обрати улюбленця, послугу, дату-час й ветеринара, до якого здійснюється запис.
- Відображення історії записів створених клієнтом. Система повинна надавати доступ клієнту до переліку створених ним записів на прийом.
- Відображення записів у чергах. Адміністратор має бачити всю інформацію про черги, позицію запису в черзі, до якого ветеринара і сам запис. Знати, які ветеринари більш затребувані і збалансовано розподіляти навантаження між ними.
- Зміна порядку запису в черзі. Адміністратор повинен мати змогу змінити порядок записів клієнтів у черзі до конкретного ветеринара.
- Переадресація запису до іншої черги. У разі потреби запис може бути перенесений до іншого ветеринара та на інший час.

- Видалення запису з черги. Адміністратор повинен мати змогу видалити запис із черги, наприклад за потребою клієнта.

Нефункціональні доповнюють функціональні вимоги та створюють передумови для стабільної роботи системи в умовах реального використання. До них входять:

- Сумісність із браузером. Інтерфейс системи повинен коректно відображатися у браузерах Chrome, Firefox, та інші.
- Швидкодія інтерфейсу. Затримка при завантаженні сторінок або обробці запитів не повинна перевищувати 2 секунд.
- Захист даних і паролів. Система повинна використовувати сучасні криптографічні засоби захисту даних.
- Доступність і надійність системи. Система повинна бути доступною не менше ніж 99% часу за рік й мати резервне копіювання.
- Інтуїтивність інтерфейсу. Користувацький інтерфейс повинен бути простим для розуміння.

Створено дворівневу діаграму ієрархії вимог на якій зображено всі 3 види вимог: бізнес, функціональні та нефункціональні вимоги системи.

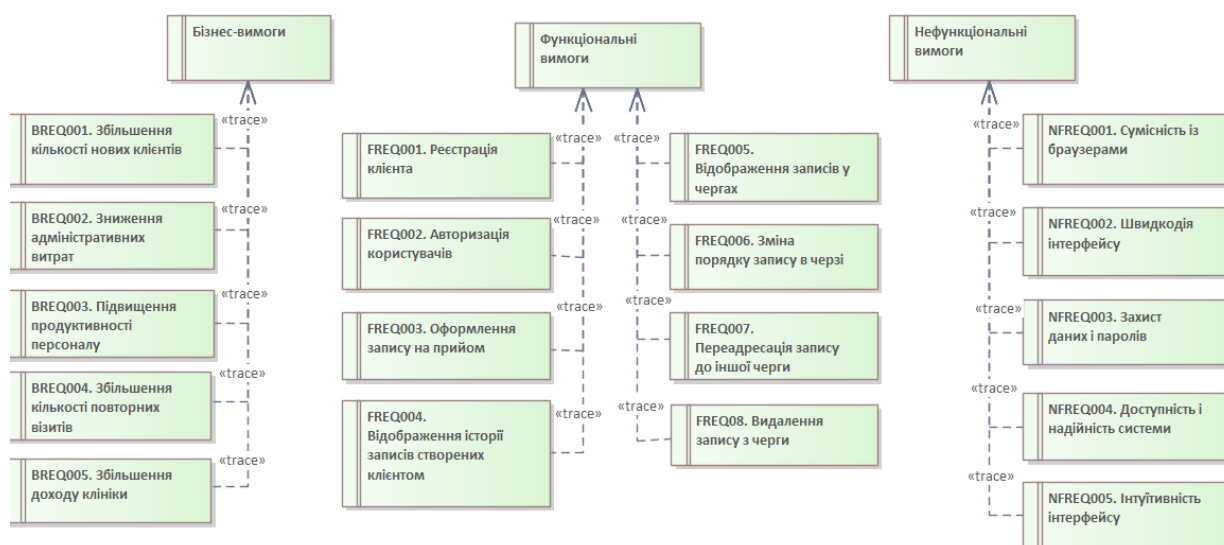


Рисунок 2.1 – Дворівнева діаграма ієрархії вимог

2.2 Постановка та алгоритм розв'язання задачі

2.2.1 Постановка задачі

Характеристика задачі. Задачею розроблення ІС є створення електронної платформи для запису та управління чергою для ветеринарної клініки, щоб можна було гарантувати автоматизацію основних процесів реєстрації клієнтів, легке створення записів на прийом до ветеринарів і адміністрування черг. Призначенням задачі є оптимізація процесу організації прийомів до клініки, полегшення взаємодії між власниками улюбленців та персоналом й збільшення загальної якості обслуговування клієнтів.

Техніко-економічна сутність лежить в основі скорочень витрат клініки на паперовий документообіг та адміністративне управління записами в черзі, а в той час автоматизація уможливлює зниження кількості помилок при введенні даних людиною, дозволяє оптимізувати навантаження на працівників клініки й робить розподіл робочого часу ветеринарів дієвішим.

Перелік об'єктів управління у системі електронного запису і управління чергою містить: клієнтів, адміністраторів клініки, облікові записи з даними для авторизації у системі. Окрім них обробляються й записи на прийом, які створюють клієнти в системі шляхом вибору параметрів улюбленця, ветеринара, послуги, дати-часу прийому, в подальшому будують чергу прийомів для кожного лікаря.

Призначення та використання вихідної інформації у системі залежить від ролі користувача в системі[25]. Клієнти через інтерфейс можуть створити обліковий запис, увійти до нього, створити запис на прийом та переглядати історію всіх створених ними записів. Адміністратори бачать список всіх записів у чергах, можуть змінювати порядок записів у черзі, переадресовувати записи до інших

ветеринарів в разі змін у розкладі або непередбачених обставин й видаляти вже неактуальні записи.

Періодичність розв'язання задачі є безперервним процесом. Всі дії користувачів миттєво відображаються в системі. Кожна зміна в черзі або в записах автоматично синхронізується, що мінімізує ризики виникнення дублювань чи некоректних даних. Вихідні дані також доступні без жодних затримок. Клієнт моментально бачить свої записи, а адміністратор зміни в черзі.

Автоматизоване розв'язання задачі припиняється у разі технічних збоїв, втрати доступу до серверу або переведення клініки на аварійний режим роботи через надзвичайні обставини.

Розподіл дій між користувачами та технічними засобами відбувається згідно з ролями: технічні засоби автоматизують процеси авторизації, запису на прийоми, управління чергою та перегляду записів[26]. Клієнти взаємодіють із системою через вебінтерфейс для створення записів, адміністратори управляють чергами і сприяють безперебійній роботі системи.

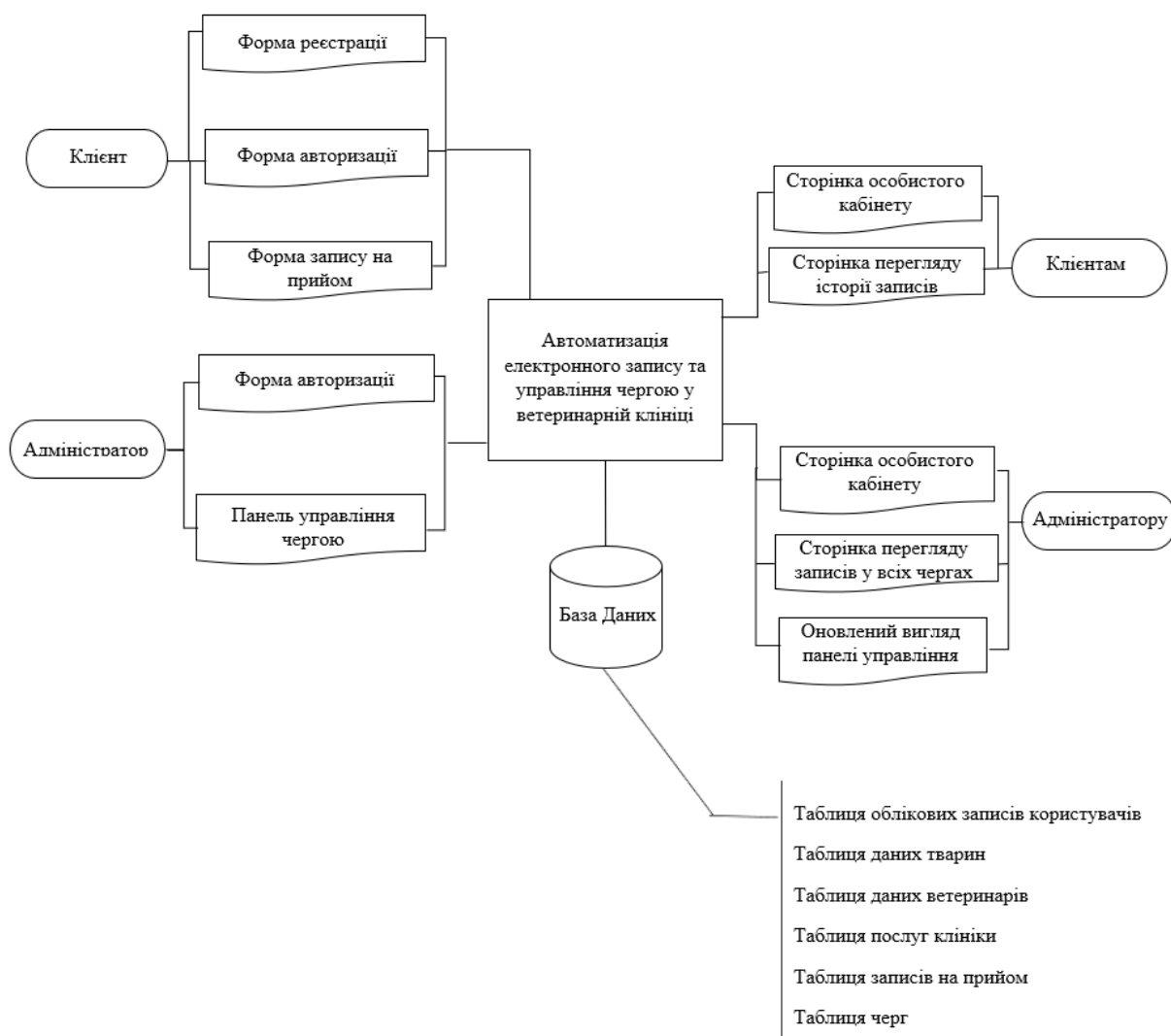


Рисунок 2.2 – Інформаційна модель розв'язання задачі електронного запису та управління чергою у ветеринарній клініці

Джерело: сформовано автором на основі виконаного дослідження

Вихідна інформація. Вихідні дані створюються вже самою системою для виконання наприклад завдань, таких як створення списку записів клієнта, створення списку записів в чергах і для розуміння стану черги в певний момент часу. Вони застосовуються для інформування клієнтів про їхню історію записів, отримань представлень про записи та управління поточним станом черг з записами.

Таблиця 2.1 – Перелік і опис вихідних повідомлень

№ з/п	Назва вихідного повідомлення	Ідентифікатор	Форма подання і вимоги до неї	Періодичність видання	Термін видання і затримка	Користувачі інформації
1	Сторінка особистого кабінету	ODI-001	Веб-сторінка	За запитом клієнта	До 1 секунди	Клієнт, Адміністратор
2	Сторінка перегляду історії записів	ODI-002	Список записів у вигляді комп. Blazor	За запитом клієнта	До 1 секунди	Клієнт
3	Сторінка перегляду записів в усіх чергах	ODI-003	Список записів у вигляді комп. Blazor	За запитом адміністратора	До 2-3 секунд	Адміністратор
4	Оновлений вигляд панелі управління	ODI-004	Список записів у вигляді комп. Blazor	За запитом адміністратора	До 2 секунд	Адміністратор

Джерело: сформовано автором на основі виконаного дослідження

Вхідна інформація. Вхідні дані в ІС ветеринарної клініки потрібні для організації реєстрацій, авторизацій, записів на прийом, відправлення певних запитів для отримання потрібного функціонала й надходить як дані від клієнтів або адміністраторів через інтерфейс системи та інших джерел у вигляді файлів.

Таблиця 2.2 – Перелік і опис вхідних повідомлень

№ з/п	Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
1	Форма реєстрації	IDI-001	Вебформа	За рішенням клієнта	Клієнт

Продовження таблиці 2.2

№ з/п	Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
2	Форма авторизації	IDI-002	Вебформа	Кожний раз при вході	Клієнт, Адміністратор
3	Форма запису на прийом	IDI-003	Вебформа	За рішенням клієнта	Клієнт
4	Панель управління чергою	IDI-004	JSON-запит через UI	За рішенням адміністратора	Адміністратор

Джерело: сформовано автором на основі виконаного дослідження

2.2.2 Алгоритм розв'язання задачі

Використовувана інформація. Потрібна для реалізації алгоритму розв'язання задачі електронного запису та управління чергою у ветеринарній клініці. Основне призначення якої у формуванні масивів даних, що використовуються для ідентифікації клієнтів, реєстрації, зміни порядку в черзі, тощо. Використання таких масивів уможливорює повну підтримку процесів відповідно до функціональних вимог інформаційної системи.

Таблиця 2.3 – Перелік масивів використовуваної інформації

Дані для реєстрації клієнта	IDS-001	10000
Дані для авторизації користувачів	IDS-002	10000

Продовження таблиці 2.3

Дані для створення запису на прийом	IDS-003	20000
Дані для зміни порядку записів	IDS-004	10000
Дані для переадресації запису	IDS-005	10000

Джерело: сформовано автором на основі виконаного дослідження

Результати розв’язання. Вихідний масив потрібен для передачі користувачеві після виконання певного запиту чи дії і зберігання даних для подальшого використання в інших підсистемах й процесах. Призначення результатної інформації в інформуванні користувачів про виконані дій, наданні доступу в систему і ще заповненні структурованих масивів даних. Всі результати мають чітку структуру для роботи системи та її відповідність обом видів вимог, функціональним і нефункціональним вимогам.

Таблиця 2.4 – Перелік масивів результатної інформації

Масив	Ідентифікатор	Максимальна кількість записів
Список користувачів	IDR-001	50000
Список тварин	IDR-002	50000
Список ветеринарів	IDR-003	1000
Список послуг	IDR-004	100
Список записів	IDR-005	50000
Список черг	IDR-006	50000

Джерело: сформовано автором на основі виконаного дослідження

Математичний опис. Для задачі електронного запису та управління чергою у ветеринарній клініці було побудовано математичну модель з описами кількісних показники, які формуються у процесі роботи ІС.

Об’єктами такої моделі виступають: клієнти, ветеринари, записи, черги.

Створений перелік допущень моделі:

- Один клієнт може мати кілька записів,
- Кожен запис відповідає одній послугі,
- Тривалість послуги відома й постійна для кожного типу,
- Дані про записи та черги зберігаються у відповідних масивах без втрат.

Формули розрахунку основних показників:

1. Формула кількості вільних слотів для запису на конкретного ветеринара(2.1):

$$F_v = T_{\text{роб}} - \sum_{i=1}^n d_i \quad (2.1)$$

в якій:

- F_v кількість доступного часу для запису в хвилинах
- $T_{\text{роб}}$ тривалість дня ветеринара,
- d_i тривалість кожного вже зареєстрованого запису i ,
- n кількість створених записів поточного дня,

2. Формула завантаженості ветеринара за день. Вона обчислюється як частка зайнятого часу від загального часу(2.2):

$$L_v = \frac{\sum_{i=1}^n d_i}{T_{\text{роб}}} \times 100\% \quad (2.2)$$

в якій:

- L_v - завантаження ветеринара у відсотках.

3. Формула кількості активних записів у черзі до ветеринара(2.3):

$$Q_v = \sum_{i=1}^m z_i \quad (2.3)$$

в якій:

- Q_v кількість всіх записів, які опинились в черзі,
- $z_i \in \{0,1\}$ - індикатор наявності активного запису,

- m - загальна кількість клієнтів, що створили запис до конкретного ветеринара.

4. Формула найпростішого потоку (2.4):

$$P(n, x) = \frac{(\lambda x)^n e^{-\lambda x}}{n!} \quad (2.4)$$

Число n подій такого потоку, що випадають на інтервал x , розподілено за законом Пуассона.

5. Формула миттєвої щільності (2.5):

$$\lambda(t) = \lim_{x \rightarrow 0} \left(\frac{M(t+x) - M(t)}{x} \right) \quad (2.5)$$

Миттєва щільність (інтенсивність) потоку дорівнює границі відношення середнього числа подій, що припадають на елементарний інтервал часу $(t, t+x)$ до довжини інтервалу часу x , коли останній прямує до нуля[0].

Розроблена модель містить важливі аспекти електронного запису: облік тривалості послуг, завантаження робітників, структуру черги. При умові правильного формування запису і реалістичних параметрів часу тривалості послуг з моделі можна отримати дієве планування розкладу й мінімізування збігів. У випадках надто великого навантаження може бути розширена моделі механізмами управління, але основні показники залишаються чинними для аналізу актуальної ситуації.

Алгоритм розв'язання задачі на ЕОМ. Написано алгоритм для реалізації функцій електронного запису та управління чергою у ветеринарній клініці, який охоплює обробку вхідних даних, перевірку коректності запиту, доступ до довідкової інформації, формування структури запису та оновлення черги. Опис логіки:

1. Ініціалізація: запуск системи, підключення до БД, перевірка конфігурації.
2. Автентифікація користувача:
 - Перевірка введених логіна і пароля.
 - У разі успіху генерація JWT-токена.
 - У разі невдачі повідомлення про помилку.
3. Формування запису на прийом:
 - Вибір улюбленця, послуги, ветеринара, дати-часу.
 - Перевірка доступності часу для обраного ветеринара.
4. Створення запису: створення об'єкта та збереження його до відповідного масиву інформації.
5. Актуалізація черги: додання запису до історії клієнта, в чергу до відповідного ветеринара і оновлення черги.
6. Формування результату: виведення повідомлення про успіх створення запису і збереження змін до БД.
7. Обробка виняткових ситуацій:
 - При відсутності зв'язку з БД процес завершується з кодом помилки.
 - При некоректно введених даних наявне повторне введення.

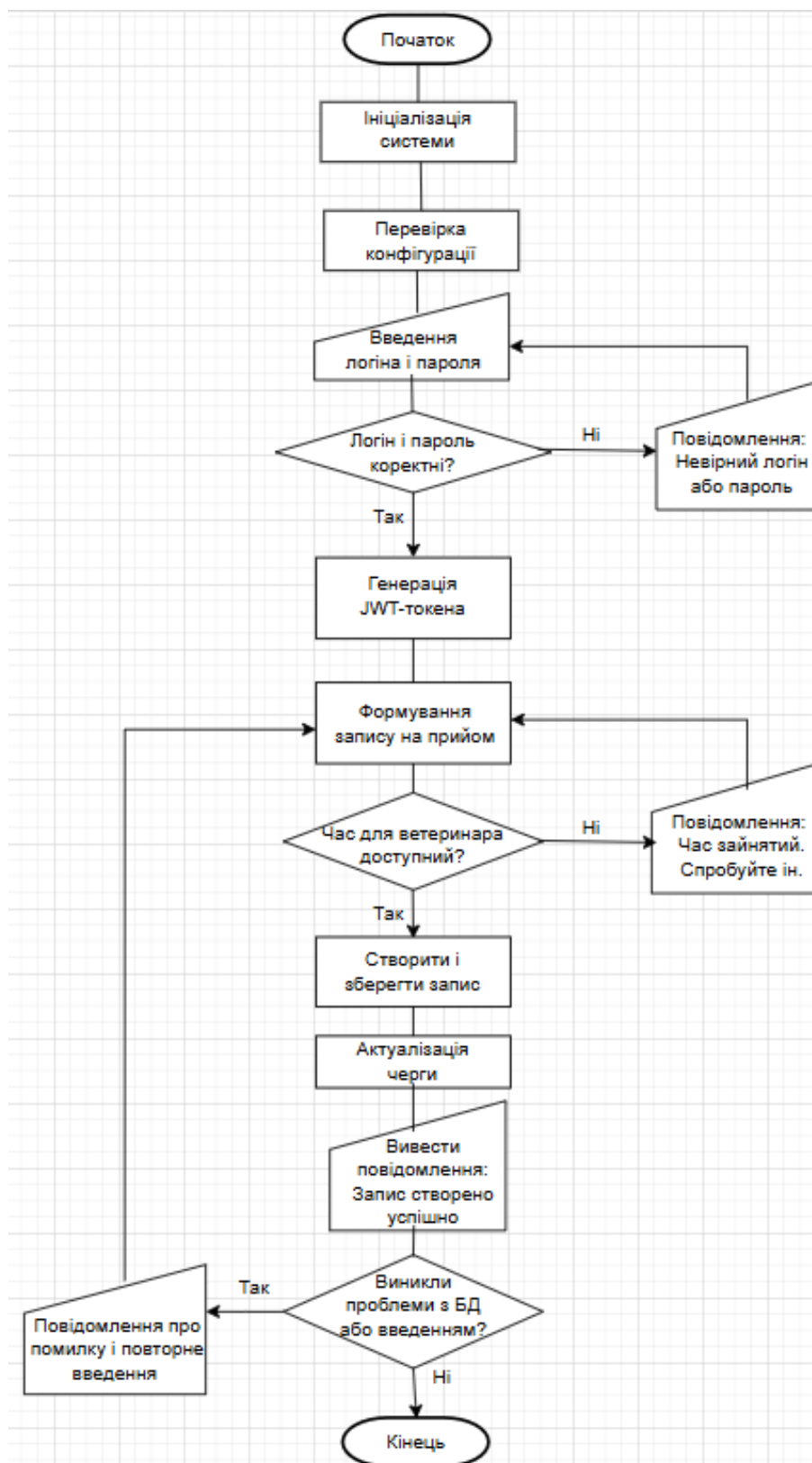


Рисунок 2.3 – Схема алгоритму задачі

Джерело: сформовано автором на основі виконаного дослідження

2.3 Моделювання інформаційної системи

2.3.1 Моделювання поведінки системи

Проектування інформаційної системи електронного запису та управління чергою для ветеринарної клініки починається з аналізу функціональних вимог. Тут формується взагалі повне уявлення про можливості системи.

Основним інструментом такого аналізу є діаграма варіантів використання, іншими словами діаграма прецедентів. З нею можна описати функціональність системи через прості сценарії взаємодій.

Наступні діаграми побудовані в середовищі Enterprise Architect.

Змодельовано діаграму прецедентів на якій наведено два актори клієнт і адміністратор, які взаємодіють з системою відповідно до своїх ролей в ній. Клієнт має можливість зареєструватися, авторизуватися, записатися на прийом, переглянути історію записів. Адміністратор, у свою чергу, може авторизуватись, переглядати записи в чергах, змінювати порядок у черзі, перенаправляти записи до інших та видаляти їх.

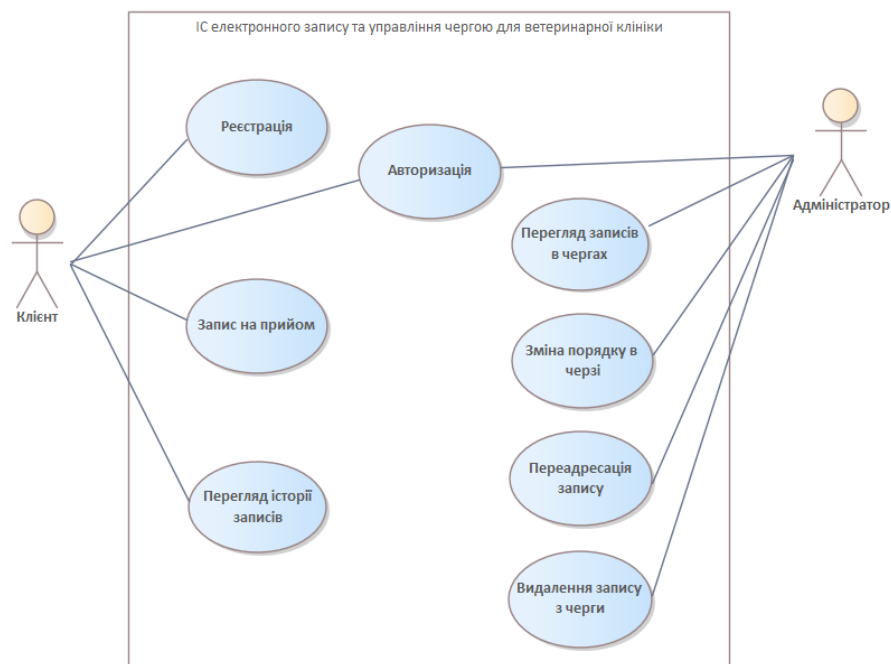


Рисунок 2.4 – Діаграма прецедентів

Джерело: сформовано автором на основі виконаного дослідження

Кожен із прецедентів зображений на діаграмі детально проаналізований шляхом побудови текстових сценаріїв, інформуючи про послідовність дій користувача та системи, іще за допомогою яких відображено логіку взаємодії між об'єктами всередині для реалізації конкретної функції. Всі сценарії мають хоча б один нормальний і відповідно до потреби, один або багато альтернативних сценаріїв. Обов'язковими елементами кожного сценарію є обмеження, передумови його початку та післяумови завершення, побудовані з допомогою розробника Scenario Builder[27].

Першим розглянемо прецедент реєстрації. Новий користувач може створити запис для подальшої роботи в системі. У нормальному сценарії клієнт відкриває форму реєстрації й вводить особисті дані як от ім'я, електронну пошту, пароль, ім'я улюбленця, вид, вік в роках, після чого система перевіряє правильність заповнення даних та унікальність email і вразі успіху після перевірки обліковий запис створюється, користувач отримує можливість перейти до авторизації.

Проте можливі ситуації коли виникає помилка під час створення запису, тому створено два альтернативні сценарії для цього прецеденту. Перший описує дії

системи і результати, у випадку коли виникла проблема з записом клієнта. Натомість другий у випадку, якщо клієнта успішно додано до БД, але є помилка з додаванням улюбленця, тоді система відкатує попередній запис клієнта, реєстрацію повністю скасовано і клієнт повертається в головне меню саме клієнта.

Type:	Scenario:		
Basic Path	Basic Path Реєстрація		
			
Step	Action	Uses	Results
1	Клієнт відкриває форму реєстрації		1. Відображається форма з полями для імені, email, пароля, улюбленця
2	Клієнт вводить дані		2. Форма приймає введені дані
3	Клієнт натискає «Зареєструватися»		3. Запит POST /api/account/register надсилається до RegistrationController
4	Система створює користувача в БД		4. Users(DB) підтверджує створення
5	Система створює улюбленця в БД		5. Pets(DB) підтверджує створення
6	Система генерує JWT-токен		6. Токен передається клієнту
7	Система перенаправляє клієнта на головну сторінку клієнта		7. Відображається головна сторінка клієнта

Рисунок 2.5 – Нормальний сценарій - Реєстрація клієнта

Джерело: сформовано автором на основі виконаного дослідження

Type:	Scenario:		
Alternate	Alternate Реєстрація - Помилка створення користувача		
			
Step	Action	Uses	Results
1	Система повертає помилку під час створення запису до таблиці Users(DB)		4.1. Виводиться повідомлення "Помилка при створенні облікового запису. Реєстрацію скасовано."
2	Система повертає клієнта до головного меню клієнта		4.2. Відображається головне меню клієнта

Рисунок 2.6 – Альтернативний сценарій №1 - Реєстрація клієнта

Джерело: сформовано автором на основі виконаного дослідження

Type:	Scenario:		
Alternate	Alternate Реєстрація - Помилка при створенні улюбленця		
			
Step	Action	Uses	Results
1	Система повертає помилку під час створення запису до таблиці Pets(DB)		5.1. Виводиться повідомлення "Помилка при спробі додати запис улюбленця. Реєстрацію скасовано."
2	Система скасовує реєстрацію		5.2. Відкат всіх змін (ROLLBACK)
3	Система повертає клієнта до головного меню клієнта		5.3. Відображається головне меню клієнта

Рисунок 2.7 – Альтернативний сценарій №2 - Реєстрація клієнта

Джерело: сформовано автором на основі виконаного дослідження

Entry Points	Context References	Constraints
Constraint	Type	Status
Обмеження - Пароль має відповідати мінімальним вимогам безпеки	Invariant	Approved
Обмеження - Унікальність логіну (електронної пошти)	Invariant	Approved
Передумова - Клієнт не має облікового запису	Pre-condition	Approved
Післямова - Створено новий обліковий запис	Post-condition	Approved

Рисунок 2.8 – Обмеження, після і передумова сценарію - Реєстрація клієнта

Джерело: сформовано автором на основі виконаного дослідження

У діаграмі до прецедента реєстрації видно обмін повідомленнями між клієнтом, формою, контролером реєстрації та таблицями БД. Зокрема, можна простежити перевірку даних та зв'язок зворотний з повідомленням про помилки або успішним створенням облікового запису.

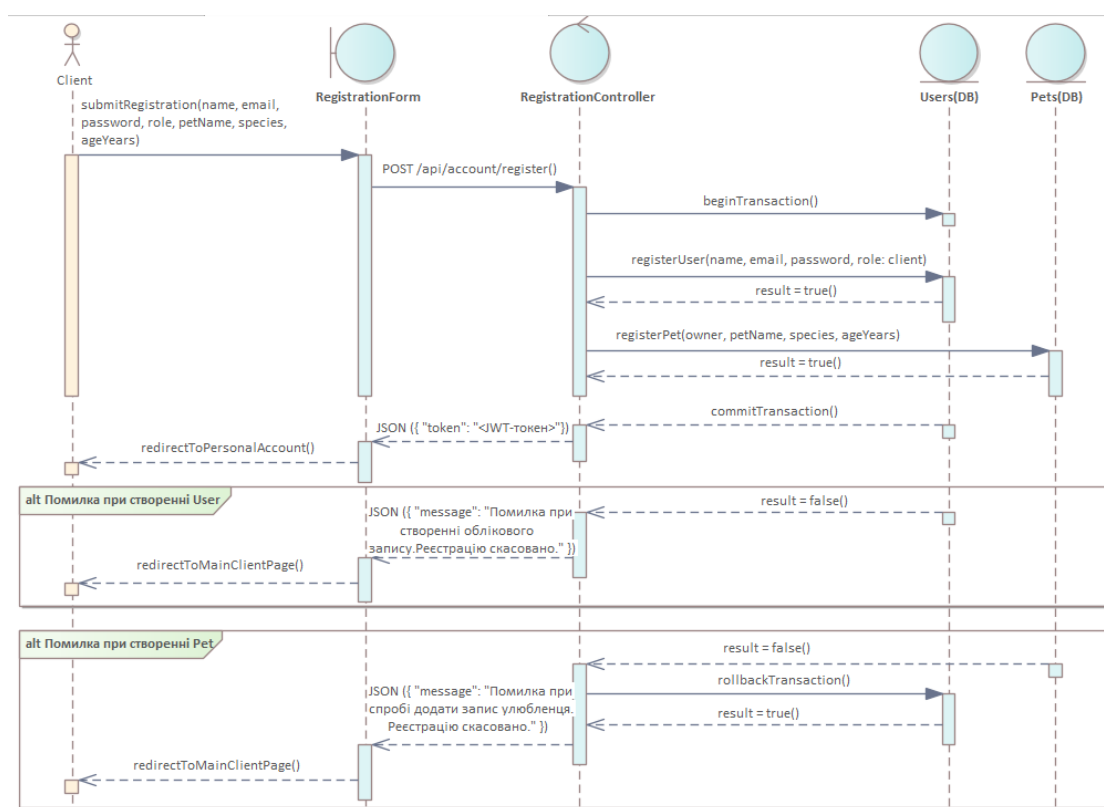


Рисунок 2.9 – Діаграма послідовності - Реєстрація клієнта

Джерело: сформовано автором на основі виконаного дослідження

Наступний прецедент авторизації відповідає за вхід обох користувачів до системи, містить також два сценарії. Нормальний починається з того, що користувач використовуючи форму авторизації вводить пошту з паролем. Після натиску на кнопку «Увійти», система перевіряє дані з обліковими записами у базі. Якщо дані правильні, перенаправляє користувача до особистого кабінету з доступом до функціоналу відповідно до його ролі. Альтернативний сценарій може виникнути у випадку, коли користувач помилково ввів невірні дані. Там система отримує помилку і повертає його в головне меню ролі користувача.

Type:	Сценарій:		
Basic Path	Basic Path Авторизація		
			
Step	Action	Uses	Results
1	Користувач відкриває форму входу.		1. Відображається форма входу
2	Користувач вводить email і пароль.		2. Дані передаються методом POST /api/account/login
3	Система перевіряє в БД.		3. Користувач знайдений, пароль співпав
4	Система надсилає JWT і роль.		4. Користувач отримує токен
5	Система перенаправляє користувача до відповідного інтерфейсу.		5. Якщо при початку роботи з системою було обрано роль <u>Клієнт</u> - перенаправлення до особистого кабінету клієнта, якщо Адмін - до особистого кабінету адміністратора

Рисунок 2.10 – Нормальний сценарій - Авторизація

Джерело: сформовано автором на основі виконаного дослідження

Type:	Scenario:		
Alternate	Alternative Авторизація - Невірний логін або пароль		
			
Step	Action	Uses	Results
 1	Система повертає помилку під час пошуку потрібного облікового запису.		3.1 Виводиться повідомлення "Неправильний email або пароль"
 2	Система повертає користувача в головне меню відповідної ролі		3.2 Відображення сторінки головного меню відповідної ролі

Рисунок 2.11 – Альтернативний сценарій №1 - Авторизація

Джерело: сформовано автором на основі виконаного дослідження

Entry Points	Context References	Constraints
Constraint	Type	Status
Обмеження - Максимум 5 спроб входу.	Invariant	Approved
Передумова - Користувач має обліковий запис.	Pre-condition	Approved
Післяумова - Користувач авторизований у системі.	Post-condition	Approved
Післяумова - Доступ до персонального кабінету відкрито.	Post-condition	Approved

Рисунок 2.12 – Обмеження, після і передумова сценарію - Авторизація

Джерело: сформовано автором на основі виконаного дослідження

Діаграма sequence показує взаємодію між користувачем, формою, контролером авторизації і таблицею Users бази даних. Візуально відображено перевірку введених даних та перенаправлення користувача до відповідного особистого кабінету в разі успіху.

Type:	Scenario:		
Alternate	Alternative Запис на прийом - Помилка при створенні запису		
Step	Action	Uses	Results
1	Система повертає помилку про спробі створити запис		3.1 Виводиться повідомлення "Помилка при створенні запису на прийом"
2	Система повертає клієнта до особистого кабінету		3.2 Відображається особистий кабінет

Рисунок 2.15 – Альтернативний сценарій №1 - Запис на прийом

Джерело: сформовано автором на основі виконаного дослідження

Entry Points	Context References	Constraints
Constraint	Type	Status
Обмеження - Неможливо записатися на зайнятий слот	Invariant	Approved
Передумова - Клієнт авторизований у системі	Pre-condition	Approved
Післяумова - Обраний слот позначено як зайнятий	Post-condition	Approved
Післяумова - Створено новий запис на прийом	Post-condition	Approved

Рисунок 2.16 – Обмеження, після і передумова сценарію - Запис на прийом

Джерело: сформовано автором на основі виконаного дослідження

На діаграмі послідовності є кроки вибору параметрів, запис у базу та зворотній зв'язок до інтерфейсів. Також враховано помилку, в разі якої система починає йти по альтернативному сценарію.

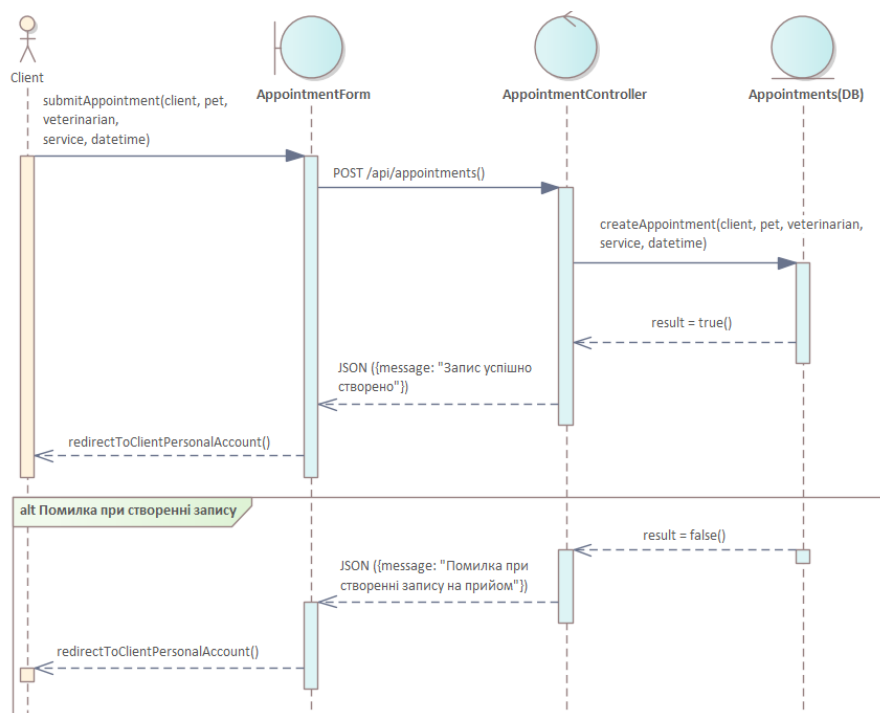


Рисунок 2.17 – Діаграма послідовності - Запис на прийом

Джерело: сформовано автором на основі виконаного дослідження

Далі текстовий сценарій перегляду історії, в якому тут наведено лише нормальний сценарій, адже потреби у іншому немає. Прецедент займається

реалізацією функцією отримання клієнтом списку всіх попередньо створених записів на прийом до ветеринара. Сценарій описує перехід клієнта на сторінку, де система формує запит до БД і повертає йому список записів із деталями.

Type:	Scenario:		
Basic Path	Basic Path Перегляд історії записів		
Step	Action	Uses	Results
1	Клієнт натискає кнопку для переходу на сторінку з історією записів		1 Робота методу GET /api/appointments/mine
2	Система вибирає записи користувача		2 Повертає список із БД
3	Система відображає історію записів		3 Клієнт бачить свої створені записи

Рисунок 2.18 – Нормальний сценарій - Перегляд історії записів

Джерело: сформовано автором на основі виконаного дослідження

Entry Points	Context References	Constraints
Constraint Type Status		
Обмеження - Історія відображається лише для власного профілю		
Pre-condition - Клієнт авторизований у системі		
Post-condition - Відображено список всіх створених записів з деталями		

Рисунок 2.19 – Обмеження, після і передумова сценарію - Перегляд історії записів

Джерело: сформовано автором на основі виконаного дослідження

У sequence діаграмі показано взаємодію між клієнтом, сторінкою, контролером історії та таблицею і звичайно ж, особливу увагу приділено запиту даних лише конкретному власнику тварин.

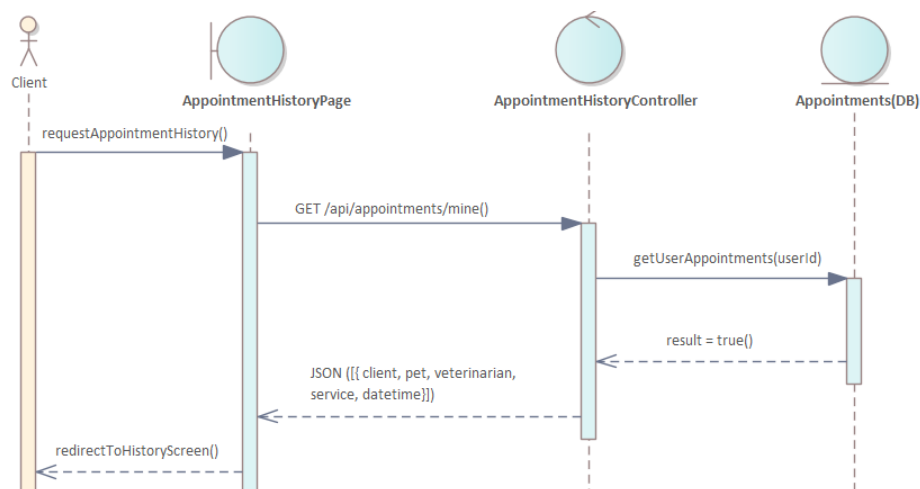


Рисунок 2.20 – Діаграма послідовності - Перегляд історії записів

Джерело: сформовано автором на основі виконаного дослідження

Як і в минулому випадку прецедент перегляду записів в чергах має тільки нормальний сценарій. Він дозволяє вже адміністратору переглядати поточний стан черг, адміністратор обирає сторінку з всіма записами, що потрапили до черг, після чого система звертається до БД і формує перелік записів.

Type:		Scenario:	
Basic Path		Basic Path Перегляд записів в чергах	
Step	Action	Uses	Results
1	Адміністратор відкриває сторінку для перегляду всіх записів з таблиць черг		1 Робота методу GET /api/queues
2	Система вибирає всі поля з таблиць Queues		2 Повертається JSON з усіма даними
3	Система відображає сторінку з усіма записами і інформацією про номер черги і позицію		3 Адміністратор бачить місце записів у чергах і статистику про ветеринара, якого обирають найчастіше

Рисунок 2.21 – Нормальний сценарій - Перегляд записів в чергах

Джерело: сформовано автором на основі виконаного дослідження

Entry Points Context References Constraints		
Constraint	Type	Status
Обмеження - інформація про записи в чергах відображаються лише на поточному ауті	Invariant	Approved
Передумова - Адміністратор авторизований у системі	Pre-condition	Approved
Післямова - Адміністратор бачить всі записи, інформацію про номер черги і позицію	Post-condition	Approved

Рисунок 2.22 – Обмеження, після і передумова сценарію - Перегляд записів в чергах

Джерело: сформовано автором на основі виконаного дослідження

На діаграмі послідовності цього прецеденту є процес запиту через інтерфейс, на отримання від БД списку всіх записів, що потрапили до черг та відображення його адміністратору.

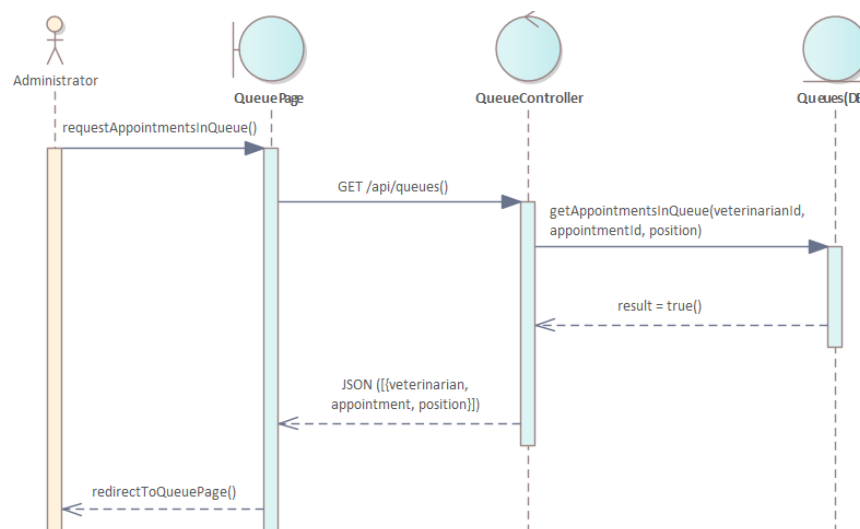


Рисунок 2.23 – Діаграма послідовності - Перегляд записів в чергах

Джерело: сформовано автором на основі виконаного дослідження

Прецедент вже зміни порядку в черзі потрібен адміністратору у випадках, коли необхідно змінити порядок відвідувачів, наприклад при запізненні, терміновому виклику іншого власника або організаційних змінах у роботі. У межах текстового сценарію він авторизується, переходить на панель управління чергами, обирає чергу якогось ветеринара з потрібним записом на прийом, змінює його позицію в черзі й переміщує вгору або вниз на інший час. Система проводить перевірку на допустимість змін на конфлікти з іншими записами, оновлює порядок записів у БД.

Type:	Scenario:		
Basic Path	Basic Path Зміна порядку записів		
			
Step	Action	Uses	Results
 1	~~Адміністратор змінює позицію запису.~~		1 Робота методу PUT /api/queues/reorder
 2	~~Система оновлює позицію в черзі~~		2 БД підтверджує оновлення
 3	~~Система відображає оновлену сторінку управління чергами~~		3 Показується актуальний список записів у чергах

Рисунок 2.24 – Нормальний сценарій - Зміна порядку запису

Джерело: сформовано автором на основі виконаного дослідження

Entry Points	Context References	Constraints
Constraint	Type	Status
Обмеження - Доступні лише вільні слоти	Invariant	Approved
Передумова - Існує активний запис на прийом	Pre-condition	Approved
Передумова - Адміністратор авторизований у системі	Pre-condition	Approved
Післяумова - Час прийому змінено на новий доступний слот	Post-condition	Approved

Рисунок 2.25 – Обмеження, після і передумова сценарію - Зміна порядку запису

Джерело: сформовано автором на основі виконаного дослідження

Діаграма для зміни порядку запису детально зображає ось цей процес. Видно початкову взаємодію адміністратора з панеллю управління і генерацією запиту сервером для зміни позиції, оновлення даних у системі та формування нового відображення черги. Відповідний варіант використання уможливорює оперативне реагування на запис, без необхідності його скасовувати.

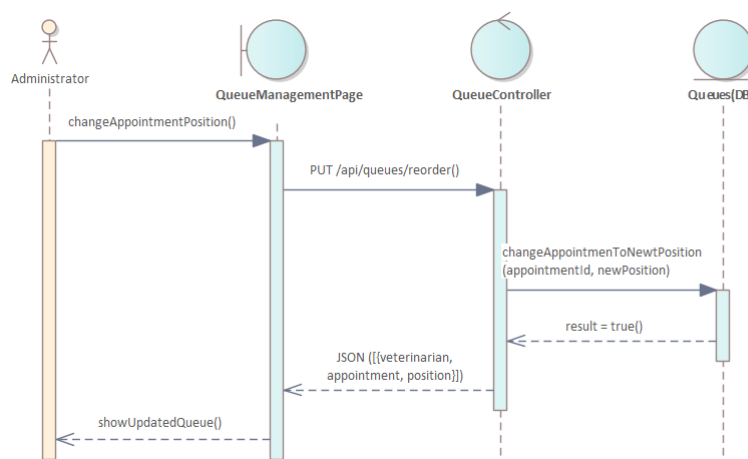


Рисунок 2.26 – Діаграма послідовності - Зміна порядку записів в черзі

Джерело: сформовано автором на основі виконаного дослідження

Ще один прецедент переадресації запису на прийом. Реалізує адміністративну функцію, переадресовує запис. Потреба виникає, якщо попередньо обраний лікар захворів, відсутній або має інші справи. У сценарії видно як адміністратор через інтерфейс обирає запис, натискає кнопку переадресації у вигляді стрілочок, потім вже вказує нове його розташування. Система перевіряє доступності і якщо все коректно, оновлює запис у двох таблицях БД.

Type:		Scenario:	
Basic Path		Basic Path Переадресація запису	
			
Step	Action	Uses	Results
1	Адміністратор вибирає запис і потрібне місце в черзі до іншого ветеринара		1 Робота методу PUT /api/appointments/redirect
2	Система оновлює поле Veterinarian в таблиці Appointments		2 Підтвердження оновлення таблиці Appointments
3	Система оновлює поле Veterinarian в таблиці Queues		3 Підтвердження оновлення таблиці Queues
4	Система відображає оновлені черги		Адміністратор бачить перенесений запис в новій черзі

Рисунок 2.27 – Нормальний сценарій - Переадресація запису

Джерело: сформовано автором на основі виконаного дослідження

Entry Points	Context References	Constraints
Constraint		
Constraint	Type	Status
Обмеження - Ветеринар повинен мати вільний час у вибрану дату	Invariant	Approved
Передумова - Існує активний запис на прийом	Pre-condition	Approved
Передумова - Адміністратор авторизований у системі	Pre-condition	Approved
Післяумова - Запис змінено і перенесено до черги іншого ветеринара	Post-condition	Approved

Рисунок 2.28 – Обмеження, після і передумова сценарію - Переадресація запису

Джерело: сформовано автором на основі виконаного дослідження

Entry Points Context References Constraints		
Constraint	Type	Status
Обмеження - Неможливо видалити вже завершені або неактивні записи	Invariant	Approved
Передумова - Існує запис який ще не відбувся	Pre-condition	Approved
Передумова - Адміністратор авторизований у системі	Pre-condition	Approved
Післяумова - Запис видалено з черги	Post-condition	Approved
Післяумова - Слот став доступним для запису на прийом	Post-condition	Approved

Рисунок 2.31 – Обмеження, після і передумова сценарію - Видалення запису

Джерело: сформовано автором на основі виконаного дослідження

На останній процес представлено крок за кроком. Спочатку взаємодія з інтерфейсом, саме з кнопкою видалення, далі обробка запиту на стороні сервера, зміни в базі даних та відображення оновлення для черги адміністратора.

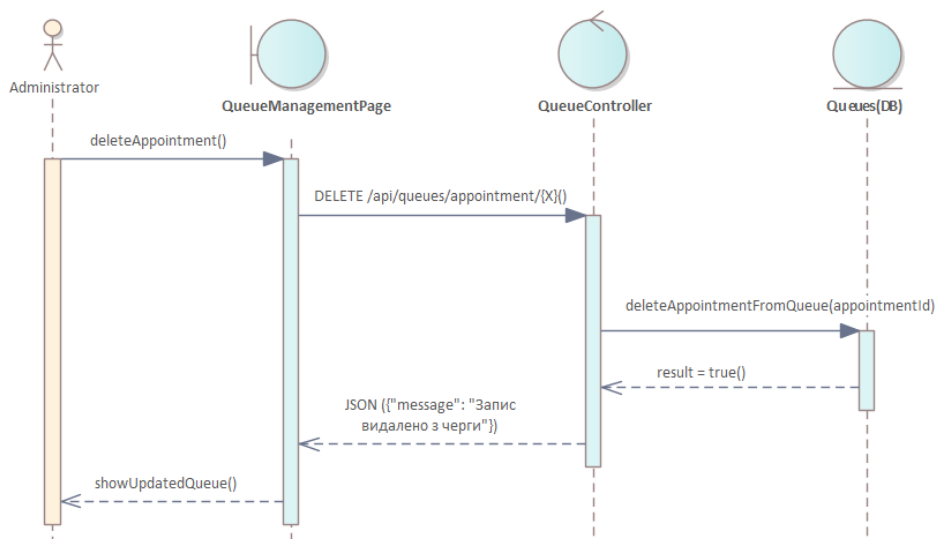


Рисунок 2.32 – Діаграма послідовності - Видалення запису

Джерело: сформовано автором на основі виконаного дослідження

2.3.2 Моделювання структури системи

Використано мову моделювання SysML, щоб можна було побудувати моделі ІС електронного запису та управління чергою для ветеринарної клініки. Структура системи розглянута за допомогою трьох різноманітних діаграм: діаграма визначення ієрархії основних блоків, їх взаємодія та потоки між ними на діаграмі

внутрішніх блоків, остання об'єктна модель у вигляді класів. Перша діаграма визначення блоків представляє ієрархію структури компонентів системи, має розділення на два головні блоки: User Devices та Server.

До блоку пристроїв входить веббраузер, через який користувач взаємодіє з системою. Він пов'язаний із фронтом Blazor UI, містить модулі презентаційної логіки та інтерфейсних елементів, таких як кнопки, сторінки, форми і т.д.

Серверний блок охоплює розділені Backend System API та Relational Database SQL Server. API представлений трьома підблоками: API клієнта, API панелі адміністратора та загальний API авторизації. Тим часом, база даних структурована на окремі таблиці, які зберігають дані про користувачів, домашніх тварин, ветеринарів, сервіси, записи та черги прийомів.

Відповідна структура має поділ системи на фронтенд, серверну логіку та базу даних, дозволяє бачити взаємозв'язки між блоками й попри все це ще й забезпечує масштабованість архітектури.

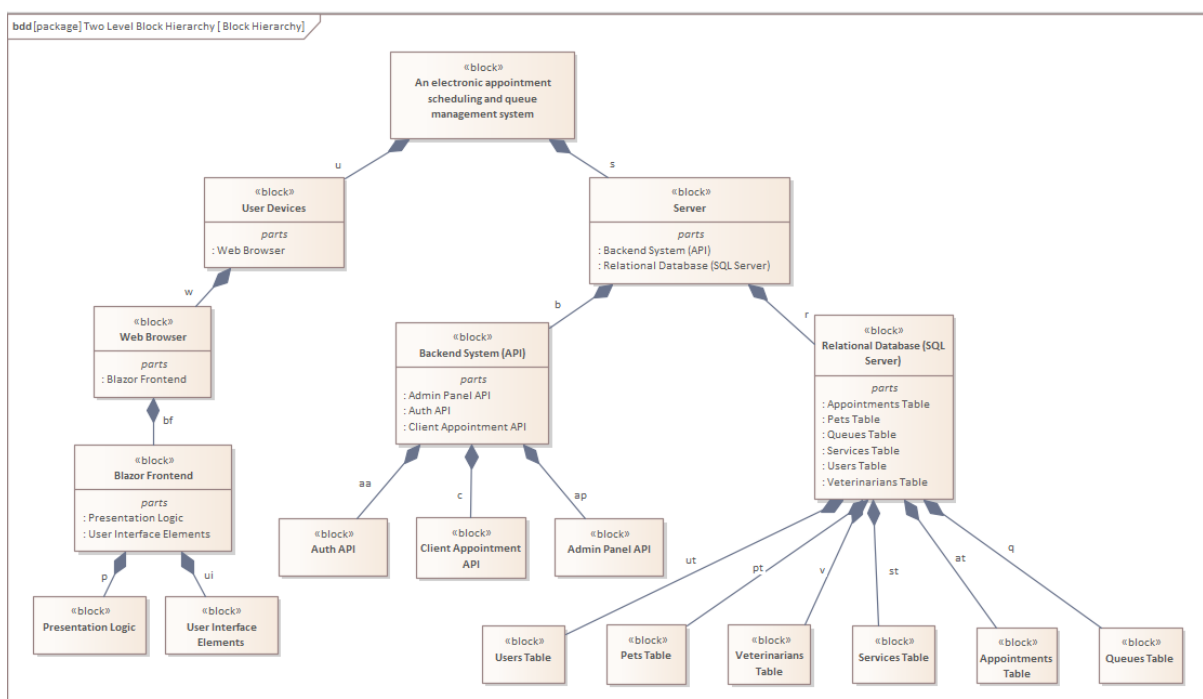


Рисунок 2.33 – Діаграма визначення блоків

Джерело: сформовано автором на основі виконаного дослідження

Після створення діаграми визначення блоків, було обрано блок Server для деталізації на діаграмі внутрішніх блоків, де представлено взаємодію внутрішніх частин серверної частини системи. У межах блоку виділено три основні частини: Web Browser, Backend System (API) та SQL Server. Вони пов'язані між собою через відповідні порти та інтерфейси обміну даними. Потоки передачі даних позначені як httpRequest, httpResponse, apiRequest, apiResponse, dataRequest і dataOutput.

Порт httpRequest приймає вхідні HTTP-запити, отримані через userRequest, і передає їх до частини Backend System. Після обробки запиту відповідь повертається через порт httpResponse, і через userResponse її отримує користувач. Тут відображено логіку обробки інтерфейсних запитів на стороні браузера саме початковий запит, очікування відповіді, її отримання та відображення користувачу.

Пара портів apiRequest й apiResponse демонструє зв'язок між браузером і серверною логікою. Порт apiRequest приймає оброблений запит від браузера, активує відповідний функціонал системи(API), після чого сформована відповідь передається назад через apiResponse.

Порти блоку бази даних dataRequest, dataOutput відіграють роль каналів доступу до неї. dataRequest приймає запити на зчитування або зміну даних від Backend System API, щоб отримати історію записів клієнтів, додати новий запис в чергу, оновити її тощо. Після обробки в БД результат повертається через dataOutput, передається назад до Backend System і потім через браузер до користувача.

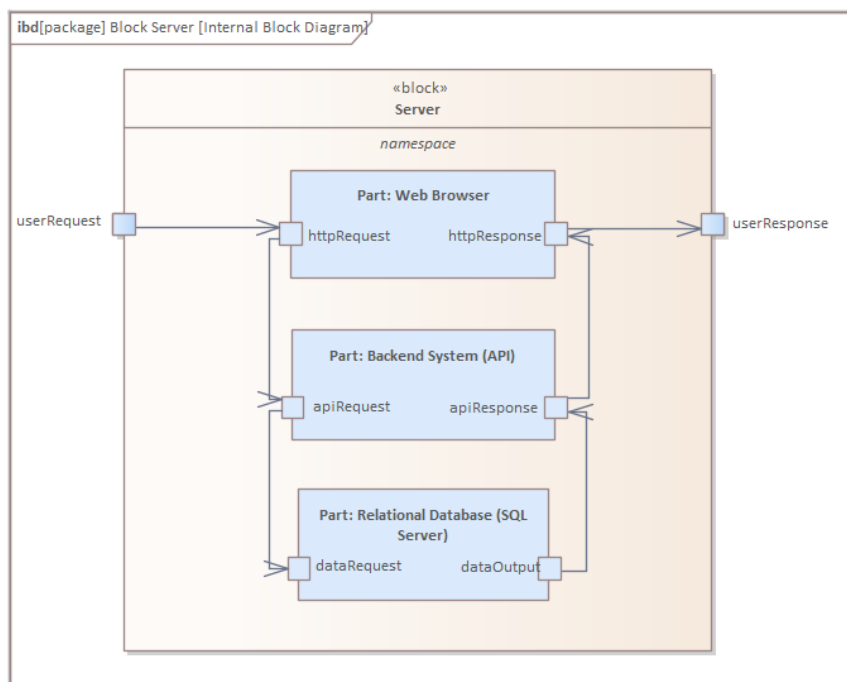


Рисунок 2.34 – Діаграма внутрішніх блоків

Джерело: сформовано автором на основі виконаного дослідження

Також, побудовано діаграму класів з поділом на ключові класи системи: граничні, керуючі та класи-сутності. Відповідний поділ на 3 видів класів сприяє чіткості архітектурного проектування й розмежує зони відповідальності компонентів.

Граничні представляють сторінки інтерфейсу: RegistrationForm, LoginForm, AppointmentForm, AppointmentHistoryPage, QueuePage, QueueManagementPanel. Кожний клас має свої атрибути, операції для взаємодії і передачу даних до контролерів.

Керуючі класи, які і є контролерами на наведених раніше діаграмах послідовностей, реалізують логіку сценаріїв прецедентів, координують взаємодію між формами інтерфейсу та об'єктами бази даних. На діаграмі представлено атрибути з операціями для: RegistrationController, AuthController, AppointmentController, AppointmentHistoryController і QueueController.

На відміну від попередніх, класи-сутності відповідають за збереження та опрацювання даних. Є наступні сутності: User, Pet, Veterinarian, Appointment,

трасування для ІС ветеринарної клініки, з ілюстрацією взаємозв'язків між вимогами, прецедентами, тест-кейсами та елементами інтерфейсу. Кожна з вимог представлена окремим елементом типу Requirement, який пов'язаний з відповідним Use Case. Сценарії підтверджуються тестом і можна відслідковувати перевірку коректності його реалізації. Ще взаємодія з елементами інтерфейсу для змоги виконувати відповідні дії в системі, дозволяє візуально оцінити, повноту реалізованої логіки в системі.

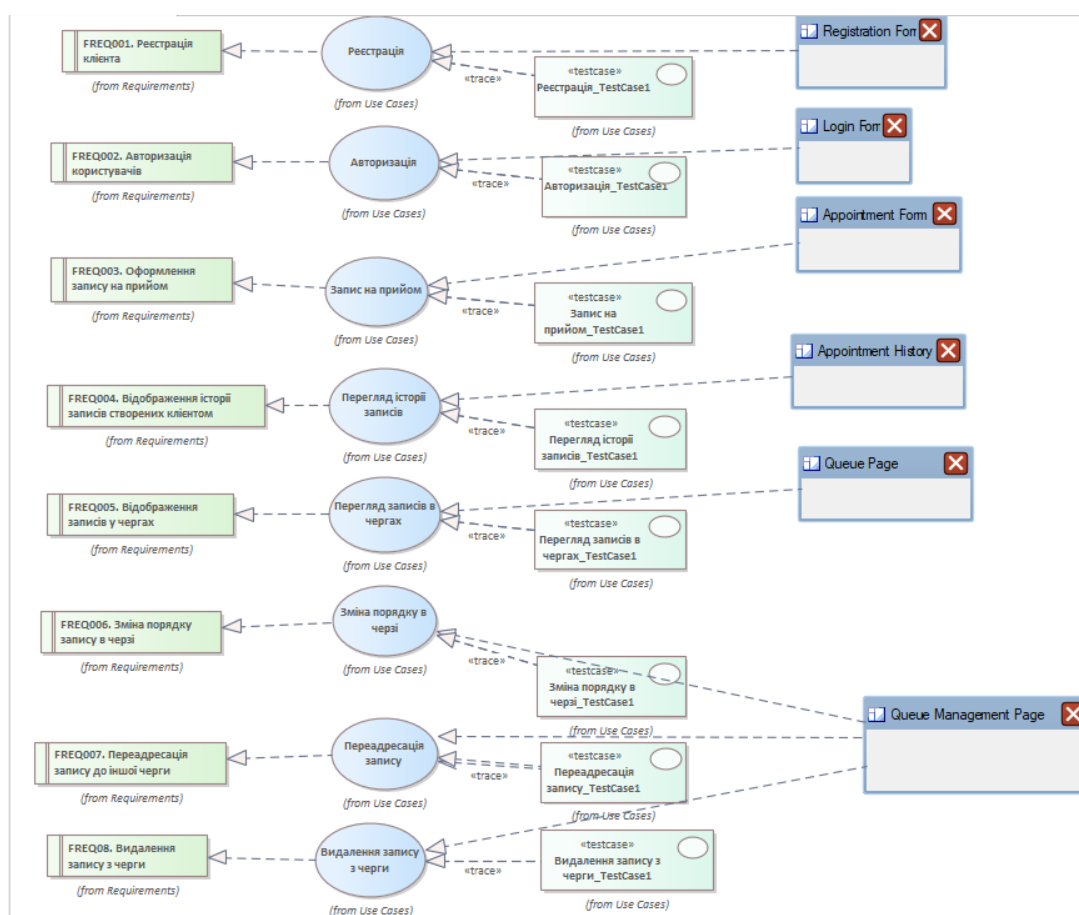


Рисунок 2.36 – Діаграма трасування

Джерело: сформовано автором на основі виконаного дослідження

Крім діаграми трасування для деталізації було сформовано матриці взаємозв'язків, у яких для кожної вимоги вказано відповідні елементи реалізації прецеденти, тести кейси та UI компоненти. Заповнено матрицю трасування прецеденти-вимоги. Можна побачити, що жодна вимога не залишилась нереалізованою.

Source \ Target	Requirements: FREQ001. Реєстрація клієнта	Requirements: FREQ002. Авторизація користувачів	Requirements: FREQ003. Оформлення запису на прийом	Requirements: FREQ004. Відображення історії записів створе	Requirements: FREQ005. Відображення записів учергах	Requirements: FREQ006. Зміна порядку запису в черзі	Requirements: FREQ007. Переадресація запису до іншої черги	Requirements: FREQ008. Видалення запису з черги
Use Cases::Авторизація	↑							
Use Cases::Видалення запису з черги								↑
Use Cases::Запис на прийом		↑						
Use Cases::Зміна порядку в черзі					↑			
Use Cases::Переадресація запису							↑	
Use Cases::Перегляд записів в чергах				↑				
Use Cases::Перегляд історії записів			↑					
Use Cases::Реєстрація	↑							

Рисунок 2.37 – Матриця трасування прецеденти - вимоги

Джерело: сформовано автором на основі виконаного дослідження

Наступна матриця трасування тести-кейси - прецеденти. З її допомогою видно як кожен прецедент має принаймні один тестовий випадок, і бути впевненим, що усі дії користувачів перевіряються під час тестування системи. Побудувавши матрицю трасування, можна вчасно виявити недоліки в логіці реалізації функціоналу. Матриця тут виступає інструментом контролю якості будови.

Source: Use Cases

Set

Type: TestCase

Link Type: Realization

Target: Use Cases

Set

Type: UseCase

Direction: Source -> Target

Source \ Target	Use Cases::Авторизація	Use Cases::Видалення запису з черги	Use Cases::Запис на прийом	Use Cases::Зміна порядку в черзі	Use Cases::Переадресація запису	Use Cases::Перегляд записів в чергах	Use Cases::Перегляд історії записів	Use Cases::Реєстрація
Use Cases::Авторизація_TestCase1	↑							
Use Cases::Видалення запису з черги_TestCase1		↑						
Use Cases::Запис на прийом_TestCase1			↑					
Use Cases::Зміна порядку в черзі_TestCase1				↑				
Use Cases::Переадресація запису_TestCase1					↑			
Use Cases::Перегляд записів в чергах_TestCase1						↑		
Use Cases::Перегляд історії записів_TestCase1							↑	
Use Cases::Реєстрація_TestCase1								↑

Рисунок 2.38 – Матриця трасування тест-кейси - прецеденти

Джерело: сформовано автором на основі виконаного дослідження

Третьою є матриця трасування елементів інтерфейсу - прецеденти, яка демонструє зв'язок між формами користувацького інтерфейсу та відповідними прецедентами. Верифікація за допомогою цього підходу дає змогу встановити, як кожен елемент інтерфейсу є необхідним і виконує конкретну функцію, яка ж відповідає одному або кільком прецедентам. Матриця допомагає знайти можливі повторення або і недоліки в дизайні, а ще зробити навігацію в системі зручною та послідовною.

Source:	Interface	Set	Type:	Screen	Link Type:	Realization
Target:	Use Cases	Set	Type:	UseCase	Direction:	Source ->
Target +						
						Use Cases: Авторизація
						Use Cases: Видалення запису з черги
						Use Cases: Запис на прийом
						Use Cases: Зміна порядку в черзі
						Use Cases: Переадресація запису
						Use Cases: Перегляд записів в чергах
						Use Cases: Перегляд історії записів
						Use Cases: Реєстрація
+ Source						
Main Menu::Main Menu						
Main Page Administrator::Main Page Administrator						
Main Page Client::Main Page Client						
Personal Administrator's Account::Personal Administrator's Ac...						
Personal Client's Account::Personal Client's Account						
User Interface Appointment History::Appointment History Page						↑
User Interface Appointment::Appointment Form						↑
User Interface Login::Login Form						↑
User Interface Queue Management ::Queue Management Page						↑
User Interface Queue Page::Queue Page						↑
User Interface Registration::Registration Form						↑

Рисунок 2.39 – Матриця трасування елементи інтерфейсу - прецеденти

Джерело: сформовано автором на основі виконаного дослідження

РОЗДІЛ 3

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

3.1 Інформаційне забезпечення

3.1.1 Загальна характеристика інформаційного забезпечення

У системі інформаційне забезпечення обробляє, збирає, зберігає та передає дані для її стабільної роботи. ІЗ охоплює логічну і фізичну організацію даних, визначає правила їхньої структури, способи доступу й методів контролю достовірності та надійності.

Структура інформаційного забезпечення включає кілька взаємопов'язаних складових. Основу становить вхідна інформація, яка надходить до системи у вигляді вебформ або запитів від користувачів і адміністраторів, наприклад реєстраційні та авторизаційні дані, запити на створення запису, запити до черги чи історії записів, запити на зміну чи видалення записів. Після обробки дані формують вихідну інформацію у вигляді повідомлень, відповідей на запити або результати авторизації, що повертаються користувачам.

В основі системи лежить база даних на платформі SQL Server, доступ до якої реалізовано за допомогою технології EF Core. Усі таблиці є фізичним носієм інформації та логічно згруповані у внутрішній інформаційній базі. Вони використовуються в процесі обробки запитів та формуванні масивів результатної інформації.

Важливий компонент ІЗ - інформаційні масиви. Масиви поділяються на використовувану та результатну інформацію. До використовуваної належать

масиви, що формуються під час взаємодії користувача з системою, тобто дані реєстрації, створення запису, переадресації тощо. А вже довідники виступають результатною інформацією, адже є даними, що система видає в відповідь на запити.

Система побудована на основі технологій ASP.NET Core Web API, Blazor і EF Core. Технології сприяють високій інтерактивності, розширюваності і легкості розроблення. Вся логіка перевірки, обробки та обміну інформацією розподілена між компонентами Blazor та контролерами ASP.NET.

Передбачено валідацію даних і використання механізмів автентифікації, авторизації та логування для достовірності й безпеки даних.

База даних SQL обрана через структуроване зберігання взаємопов'язаних сутностей, простотою масштабування, підтримкою транзакцій та широким інструментарієм ORM доступу через EF Core. Для системи ветеринарної клініки вона є найкращим вибором гарантує її безперебійність.

Циркулююча інформація у системі має чітко визначені джерела, формати, способи подання й час реакції. Система працює з мінімальними затримками та високим рівнем достовірності, значуще обслуговування власників тварин в момент події.

Подана схема інформаційного забезпечення системи з поділом на основні блоки: методичні і інструментальні матеріали, система класифікації і кодування, інформаційна база внутрішня й зовнішня.

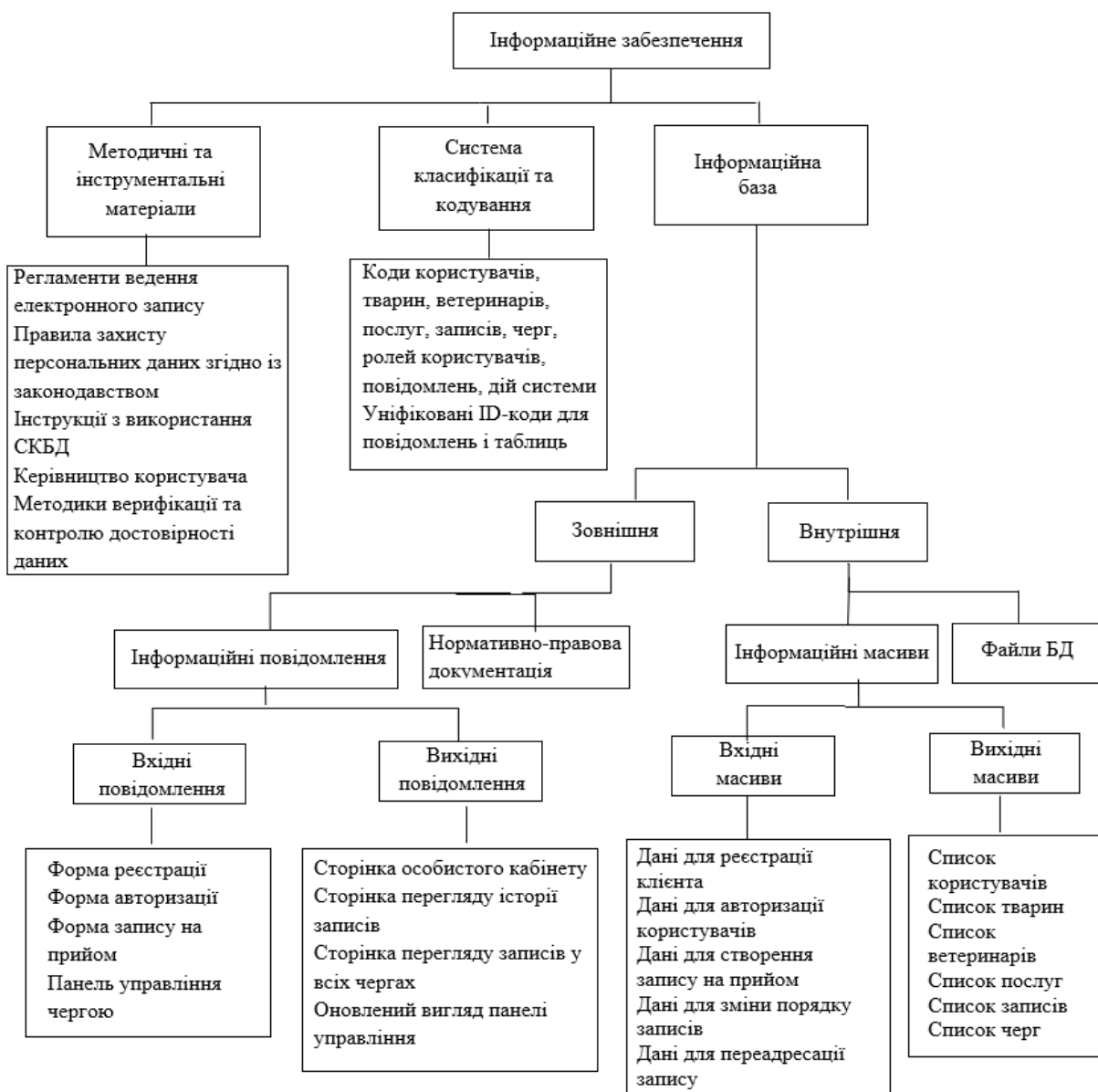


Рисунок 3.1. Схема ІЗ системи

Джерело: сформовано автором на основі виконаного дослідження

3.1.2 Організація збору і передавання первинної інформації

В інформаційній системі автоматизації електронного запису та управління чергою для ветеринарної клініки інформація надходить на обробку в момент запуску виконання дій від клієнта або адміністратора.

Користувачі, а саме клієнт і адміністратор, база даних ветеринарної клініки є інформацією основних джерел. Ролі користувачів існують для ініціювання передавання інформації через інтерфейс в системі.

Повідомлення надходять до системи через вебінтерфейс у формі запитів JSON або форм, після автоматично передаються на обробку серверу.

Кожне вхідне повідомлення створюється роботою користувача в браузері та потрапляє в систему для формування відповідної відповіді. Можна виділити такі повідомлення в системі ветеринарної клініки: форми реєстрації/авторизації, форма створення запису, панель управління чергою тощо.

Носії первинної інформації:

- Файли бази даних
- JSON-повідомлення
- Система керування базами даних СКБД
- Web API сервер додатку

Інформація зберігається в внутрішній базі даних. Періодичність надходження залежить від користувача. Способом передачі слугує мережева передача JSON структур по HTTP/HTTPS протоколу. Первинні дані записуються на електронні носії в структурованій таблиць SQL формі. Вона потім опрацьовується для наступних етапів, наприклад при генерації вихідних повідомлень й результатів розв'язання.

3.1.3 Побудова системи класифікації та кодування

Система класифікації в інформаційній системі ветеринарної клініки має структурований підхід до організації даних. Всі ключові об'єкти системи поділяються на користувачів, тварин, ветеринарів, послуг, записів на прийом, черг. Всі категорії користуються унікальною схемою ідентифікації та кодування для безпомилкового розпізнання об'єктів і хуткого виконання пошуку, сортувань й зв'язувань. Далі наведені типи кодів системи:

- Класифікаційні коди об'єктів: ID User, ID Pet, ID Veterinarian, ID Service, ID Appointment, ID Queue.
- Коди вхідних й вихідних повідомлень: IDI, ODI.
- Коди вхідних й вихідних масивів даних: IDS, IDR.

Для кодування всіх типів кодів обрано метод символно-цифровий.

Структурою коду для об'єктів таблиць є 3 літери коду категорії об'єкту і 5 цифр порядкового номеру, тобто довжина 8 символів. Структурою для масивів і повідомлень є 3 літери коду категорії об'єкту і 3 цифри порядкового номеру, кількість 6 символів. Спосіб сприяє запобігти дублювань та конфліктів даних, спрощує міжсистемну інтеграцію та прискорює обробку інформації на сервері.

Тобто коди виглядають як:

- IDU-00001, IDV-00002 для об'єктів таблиць,
- IDI-002, ODI-003 для інформаційних повідомлень,
- IDS-001, IDR-002 для масивів інформації.

Коди збережені в таблицях бази даних. Їх використовують в логіці запитів створення відповідей та зберіганні даних. Формуються автоматично в серверній частині системи, мають стабільну довжину, тому можна формувати посилання, проводити масові операції з даними та тримати однозначність кожного об'єкта системи.

3.1.4 Проєктування форм первинних документів та відеокадрів

Вирішено використати кілька форм первинних документів для реалізації роботи основних функції системи ветеринарної клініки. До форм первинних належать вхідні і вихідні повідомлення, результати розв'язання задач, представлені у вигляді машинограм, а саме виводи на інтерфейс користувача.

Первинні документи/форми:

1. Форма реєстрації клієнта. Заповнюється лише клієнтом при створенні облікового запису. Має на увазі введення імені, пошти, паролю та інформації про тварину. Запитом JSON надсилається через інтерфейс, клієнт - джерело. Документ розрахований на одне застосування при реєстрації.

2. Форма авторизації. Пошту і пароль вводить клієнт або вже адміністратор для отримання входу в систему. Кінцем авторизації є надання JWT токена.

3. Форма створення запису на прийом. Формується після входу клієнта і переходу на форму запису. У формі є вибір дати, ветеринара, послуги та улюбленця. Після виконання запис потрапляє в конкретну чергу.

4. Панель управління чергою, зміна порядку, переадресація або видалення запису. Утворюються як JSON команди з значенням ID запису та дією.

Вихідні документи/форми:

1. Сторінка особистого кабінету. Показує профіль з інформацією користувача.

2. Сторінка перегляду історії записів. Звичайний список минулих записів, є машинограмою, а джерелом база даних.

3. Сторінка перегляду записів у всіх чергах. Адміністратор отримує за запитом для перегляду записів з черг.

4. Оновлений вигляд панелі управління чергою. Показується адміністратору в вигляді 3 таблиць з всіма записами.

Форми подання даних реалізовані як частина інтерфейсу. Через HTML елементи з передачею запитів формату JSON здійснюється збір з обробкою інформації. Вихідна виводиться у вигляді таблиць, документи створюються майже вмість десь 1-3 секунди.

3.1.5 Структура інформаційних масивів

Написано структуру основних інформаційних масивів в системі електронного запису та управління чергою для ветеринарної клініки. Склад кожного масиву чітко регламентований і поданий у вигляді таблиці.

1. Дані для реєстрації клієнта.

Найменування - IDS-001, ідентифікатор REG_CL, носій Users, максим. об'єм 10000 записів, довжина 102 символи, метод організації реляційна таблиця, ключ UserId, ідентифікатор індексного масиву IDX_Users_Email.

Таблиця 3.1 – Структура масиву дані для реєстрації клієнта

Найменування	Ідентифікатор у програмі	Позн. в формулах	Формат	Бізнес-правила				Логічні та семантичні зв'язки
				Первин/вторинний ключ	Умова значення	Обов'язкове	Індекс	
Ім'я	client_name	-	nvarchar (30)	-	Не порожнє	Так	ІДД	-
Пошта	email	-	nvarchar (20)	-	Унікальне	Так	ІНД	-
Пароль	passwordHash	-	nvarchar (10)	-	Не порожнє	Так	ІДД	-

Джерело: сформовано автором на основі виконаного дослідження

Продовження таблиці 3.1

Найменування	Ідентифікатор у програмі	Позн. в формулах	Формат	Бізнес-правила				Логічні та семантичні зв'язки
				Первин/вторинний ключ	Умова значення	Обов'язкове	Індекс	
Ім'я улюбленця	pet_name	-	nvarchar (30)	-	Не порожнє	Так	ІДД	-
Вид	species	-	nvarchar (10)	-	Вибір з можливих	Так	ІНД	-
Рік в роках	age_years	-	int(2)	-	Цілі числа	Ні	ІДД	-

Джерело: сформовано автором на основі виконаного дослідження

2. Дані для авторизації користувачів.

Найменування - IDS-002, ідентифікатор AUTH_U, носій таблиця Users, максим. об'єм 10000 записів, довжина 40 символів, метод організації пошук унікального індекса, ключ Email, ідентифікатор індексного масиву IDX_Users_Email.

Таблиця 3.2 – Структура масиву дані для авторизації користувачів

Найменування	Ідентифікатор у програмі	Позн. в формулах	Формат	Бізнес-правила				Логічні та семантичні зв'язки
				Первин/вторинний ключ	Умова значення	Обов'язкове	Індекс	
Пошта	email	-	nvarchar (20)	-	Існуюча пошта	Так	ІНД	-
Пароль	password	-	nvarchar (20)	-	Існуючий пароль	Так	ІДД	-

Джерело: сформовано автором на основі виконаного дослідження

3. Дані для створення запису на прийом.

Найменування - IDS-003, ідентифікатор NEW_APP, носій Appointments, максим. об'єм 20000 записів, довжина 44 символи, метод організації реляційна

таблиця зовн. ключі, ключ AppointmentId, ідентифікатор індексного масиву IDX_Appointments_ClientId.

Таблиця 3.3 – Структура масиву дані для створення запису на прийом

Найменування	Ідентифікатор у програмі	Позн. в формулах	Формат	Бізнес-правила				Логічні та семантичні зв'язки
				Первин/вторинний ключ	Умова значення	Обов'язкове	Індекс	
Id клієнта	clientId	-	int(5)	FK	З таблиці Users	Так	ІНД	Users
Id тварини	petId	-	int(5)	FK	З таблиці Pets	Так	ІНД	Pets
Id ветеринара	veterinarianId	-	int(4)	FK	З таблиці Veterinarians	Так	ІНД	Veterinarians
Id послуги	serviceId	-	int(3)	FK	З таблиці Service	Так	ІНД	Service
Дата-час	date_time	-	datetime 2(7)	-	Вибір з можливих	Так	ІНД	-

Джерело: сформовано автором на основі виконаного дослідження

4. Дані для зміни порядку записів.

Найменування - IDS-004, ідентифікатор RDR_QUE, носій Queues, максим. об'єм 10000 записів, довжина 7 символів, метод організації оновлення QueuePosition, ключ QueueId, ідентифікатор індексного масиву IDX_Queues_QueuePosition.

Таблиця 3.4 – Структура масиву дані для зміни порядку записів

Найменування	Ідентифікатор у програмі	Позн. в формулах	Формат	Бізнес-правила				Логічні та семантичні зв'язки
				Первин/вторинний ключ	Умова значення	Обов'язкове	Індекс	
Id запису	appointmentId	-	int(5)	FK	З таблиці Queues	Так	ІНД	Queues
Нова позиція запису	newPosition	-	int(2)	-	Вибір з наявних	Так	ІНД	-

Джерело: сформовано автором на основі виконаного дослідження

5. Дані для переадресації запису.

Найменування - IDS-005, ідентифікатор RDT_APP, носій Appointments, об'єм 10000 записів, довжина 11 символів, метод оновлення поля VeterinarianId, ключі AppointmentId, ідентифікатор інд. масиву IDX_Appointments_VeterinarianId.

Таблиця 3.5 – Структура масиву дані для переадресації запису

Найменування	Ідентифікатор у програмі	Позн. в формулах	Формат	Бізнес-правила				Логічні та семантичні зв'язки
				Первин/вторинний ключ	Умова значення	Обов'язкове	Індекс	
Id запису	appointmentId	-	int(5)	РК, FK	З Appointments і Queues	Так	ІНД	Appointments і Queues
Новий ветеринар	veterinarianId	-	int(4)	РК	З таблиці Veterinarians	Так	ІНД	Veterinarians
Нова позиція запису	newPosition	-	int(2)	-	Вибір з наявних	Так	ІНД	-

Джерело: сформовано автором на основі виконаного дослідження

6. Список користувачів.

Найменування - IDR-001, ідентифікатор LIST_USERS, носій Users, максим. об'єм 50000 записів, довжина 85 символів, метод організації вибірка всіх, ключ UserId, ідентифікатор індексного масиву IDX_Users.

Таблиця 3.6 – Структура масиву список користувачів

Найменування	Ідентифікатор у програмі	Позн. в формулах	Формат	Бізнес-правила				Логічні та семантичні зв'язки
				Первин/вторинний ключ	Умова значення	Обов'язкове	Індекс	
Id клієнта	clientId	-	int(5)	РК	Унікальне	Так	ІНД	Users
Ім'я	client_name	-	nvarchar (30)	-	Не порожнє	Так	ІДД	-
Пошта	email	-	nvarchar (20)	-	Унікальне	Так	ІНД	-
Пароль	passwordHash	-	nvarchar (20)	-	Не порожнє	Так	ІДД	-
Роль	role	-	nvarchar (10)	-	Клієнт/адміністратор	Так	ІНД	-

Джерело: сформовано автором на основі виконаного дослідження

7. Список тварин.

Найменування - IDR-002, ідентифікатор LIST_PETS, носій Pets, максим. об'єм 50000 записів, довжина 52 символи, метод організації вибірка по OwnerId, ключ PetId, ідентифікатор індексного масиву IDX_Pets.

Таблиця 3.7 – Структура масиву список тварин

Найменування	Ідентифікатор у програмі	Позн. в формулах	Формат	Бізнес-правила				Логічні та семантичні зв'язки
				Первин/вторинний ключ	Умова значення	Обов'язкове	Індекс	
Id тварини	petId	-	int(5)	РК	Унікальне	Так	ІНД	Pets
Id клієнта	clientId	-	int(5)	FK	Унікальне	Так	ІНД	Users
Ім'я улюбленця	pet_name	-	nvarchar (30)	-	Не порожнє	Так	ІДД	-
Вид	species	-	nvarchar (10)	-	Вибір з можливих	Так	ІНД	-
Рік в роках	age_years	-	int(2)	-	Цілі числа	Ні	ІДД	-

Джерело: сформовано автором на основі виконаного дослідження

8. Список ветеринарів.

Найменування - IDR-003, ідентифікатор LIST_VETS, носій Veterinarians, максим. об'єм 1000 записів, довжина 65 символів, метод організації вибірка всіх, ключ VeterinarianId, ідентифікатор індексного масиву IDX_Veterinarians.

Таблиця 3.8 – Структура масиву список ветеринарів

Найменування	Ідентифікатор у програмі	Позн. в формулах	Формат	Бізнес-правила				Логічні та семантичні зв'язки
				Первин/вторинний ключ	Умова значення	Обов'язкове	Індекс	
Id ветеринара	veterinarianId	-	int(4)	РК	Унікальне	Так	ІНД	Veterinarians
Ім'я	veterinarian_name	-	nvarchar (30)	-	Не порожнє	Так	ІДД	-

Продовження талиці 3.8

Найменування	Ідентифікатор у програмі	Позн. в формулах	Формат	Бізнес-правила				Логічні та семантичні зв'язки
				Первин/вторин ключ	Умова значення	Обов'язкове	Індекс	
Спеціалізація	specialization	-	nvarchar (30)	-	-	Ні	ІНД	-
Доступність	available	-	bit 0,1	-	0 або 1, default 1	Так	ІНД	-

Джерело: сформовано автором на основі виконаного дослідження

9. Список послуг.

Найменування - IDR-004, ідентифікатор LIST_SRVC, носій Services, максим. об'єм 100 записів, довжина 55 символів, метод організації вибірка всіх, ключ ServiceId, ідентифікатор індексного масиву IDX_Services.

Таблиця 3.9 – Структура масиву список послуг

Найменування	Ідентифікатор у програмі	Позн. в формулах	Формат	Бізнес-правила				Логічні та семантичні зв'язки
				Первин/вторин ключ	Умова значення	Обов'язкове	Індекс	
Id послуги	serviceId	-	int(3)	РК	Унікальне	Так	ІНД	Service
Назва	name	-	nvarchar (30)	-	Не порожнє	Так	ІНД	-
Ціна	price	-	decimal(10,2)	-	Додатні числа	Так	ІНД	-
Валюта	currency	-	nvarchar (10)	-	Гривні, долари	Ні	ІНД	-

Джерело: сформовано автором на основі виконаного дослідження

10. Список записів.

Найменування - IDR-005, ідентифікатор LIST_APP, носій Appointments, максим. об'єм 50000 записів, довжина 49 символів, метод організації фільтрація ClientId або VeterinarianId, ключ AppointmentId, ідентифікатор індексного масиву IDX_Appointments.

Таблиця 3.10 – Структура масиву список записів

Найменування	Ідентифікатор у програмі	Позн. в формулах	Формат	Бізнес-правила				Логічні та семантичні зв'язки
				Первин/вторинний ключ	Умова значення	Обов'язкове	Індекс	
Id запису	appointmentId	-	int(5)	РК	Унікальне	Так	ІНД	Appointments
Id клієнта	clientId	-	int(5)	FK	З таблиці Users	Так	ІНД	Users
Id тварини	petId	-	int(5)	FK	З таблиці Pets	Так	ІНД	Pets
Id ветеринара	veterinarianId	-	int(4)	FK	З таблиці Veterinarians	Так	ІНД	Veterinarians
Id послуги	serviceId	-	int(3)	FK	З таблиці Service	Так	ІНД	Service
Дата-час	date_time	-	datetime 2(7)	-	Вибір з можливих	Так	ІНД	-

Джерело: сформовано автором на основі виконаного дослідження

11. Список черг.

Найменування - IDR-006, ідентифікатор LIST_QUE, носій Queues, максим. об'єм 50000 записів, довжина 26 символів, метод організації вибірка VeterinarianId, ключ QueueId, ідентифікатор індексного масиву IDX_ Queues.

Таблиця 3.11 – Структура масиву список черг

Найменування	Ідентифікатор у програмі	Позн. в формулах	Формат	Бізнес-правила				Логічні та семантичні зв'язки
				Первин/вторинний ключ	Умова значення	Обов'язкове	Індекс	
Id черги	queueId	-	int(5)	РК	Унікальне	Так	ІНД	Queue
Id ветеринара	veterinarianId	-	int(4)	FK	З таблиці Veterinarians	Так	ІНД	Veterinarians
Id запису	appointmentId	-	int(5)	FK	З таблиці Appointments	Так	ІНД	Appointments
Позиція	position	-	int(2)	-	Вибір з можливих	Так	ІНД	-
Статус	status	-	nvarchar (10)	-	-	Ні	ІДД	-

Джерело: сформовано автором на основі виконаного дослідження

3.1.6 Вибір СКБД

Базою даних для системи електронного запису та управління чергою для ветеринарної клініки вибрано Microsoft SQL Server. Її вибір був чітко аргументований завдяки еластичній роботі з великими обсягами даних, наявними засобами адміністрування через SQL Management Studio 2022, а ще інтеграцію з іншими продуктами тієї ж компанії.

На етапі проєктування, враховувалися вимоги до нормалізації БД, цілісності через первинні та зовнішні ключі і підготовка бази в майбутньому для масштабування. Обмеженнями стали дотримання оптимальної структури таблиць й налаштування прав доступу для підтримки безпеки даних.

3.1.7 Інфологічна модель бази даних

Створено логічну модель БД у програмі AllFusion Erwin Data Modeler.

Erwin Data Modeler - комп'ютерне програмне забезпечення для моделювання даних[28].

Модель складається з таблиць: Users, Pets, Veterinarians, Services, Appointments і Queues. Таблиці пов'язані між собою через ключові поля. Модель побудована відповідно до принципів нормалізації.

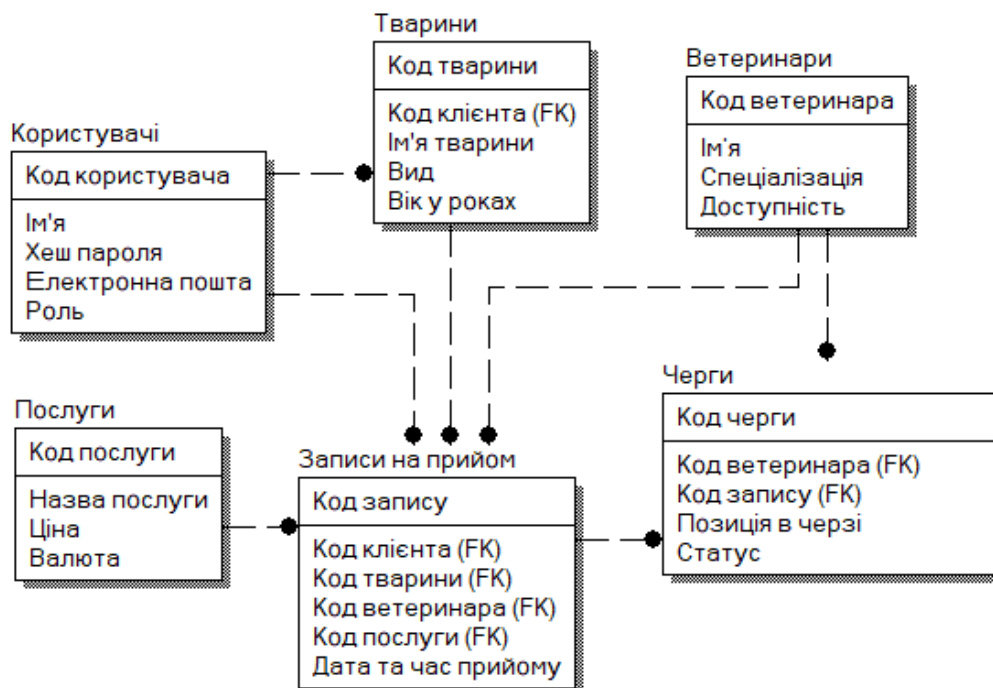


Рисунок 3.2. Логічна модель бази даних

Джерело: сформовано автором на основі виконаного дослідження

3.1.8 Даталогічна модель бази даних

Фізична модель БД створена у SQL Server. Клієнт і адміністратор зберігаються в єдиній таблиці Users. Для визначення доступу до різного функціоналу використовується поле Role. Залежно від цього значення інтерфейс програми надає або обмежує доступ до певних опцій. Відповідно до чого клієнт може зареєструватись, записатись на прийом, а адміністратор має змогу управляти чергою.

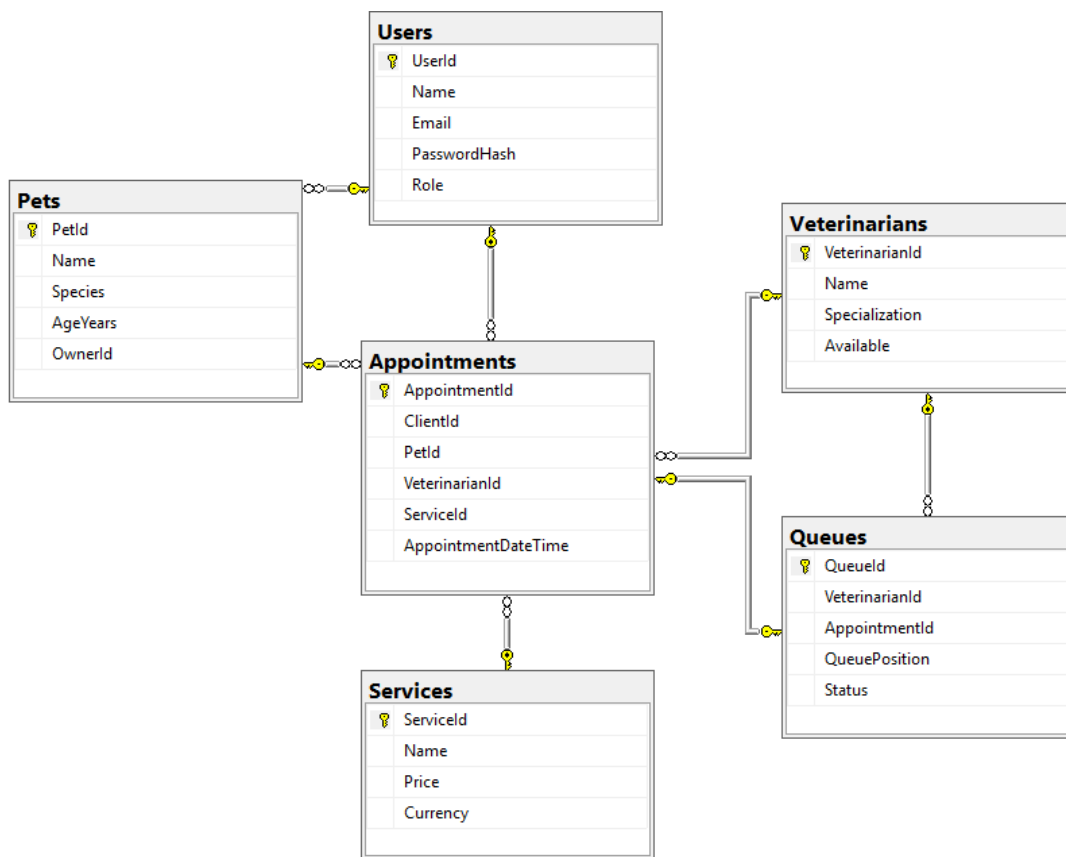


Рисунок 3.3. Фізична модель бази даних

Джерело: сформовано автором на основі виконаного дослідження

Опис складових бази в термінах DDL наведено в Додатку Б.

Далі видно первинні таблиці бази. Структуру одобрено ось приклади:

Yukie\LOCALDB#15...temDb - dbo.Users		Yukie\LOCALDB#15...stemDb - dbo.Pets		Yukie\LOCALDB#15...db	
UserId	Name	Email	PasswordHash	Role	
12	Аліна	abc@gmail.com	AQAAAAIAAYagAAAAECvYzJb6dBKG1W1mS...	Client	
14	Роза	f@gmail.com	AQAAAAIAAYagAAAAELmn6Y9cs7zpqkzBae/...	Client	
15	Дмитро	fg@gmail.com	AQAAAAIAAYagAAAAEMhjl0+oiprgpOU6B3p...	Client	
16	Сергій	fg1@gmail.com	AQAAAAIAAYagAAAAECwMKYJuq46YgQ81...	Client	
17	Вероніка	fg12@gmail.com	AQAAAAIAAYagAAAAEJDrHqHcqkyBZmeFH...	Client	
18	Аліна	fg123@gmail.c...	AQAAAAIAAYagAAAAEHwIGQQ29/GnD5UpA...	Client	
NULL	NULL	NULL	NULL	NULL	

Рисунок 3.4. Заповнені дані Users

Джерело: сформовано автором на основі виконаного дослідження

Yukie\LOCALDB#15...stemDb - dbo.Pets					
	PetId	Name	Species	AqeYears	OwnerId
▶	12	Джей	Кіт	1	12
	14	Мяу	Кіт	3	14
	15	Джейк	Хом'як	1	15
	16	Королева	Папуга	5	16
	17	Пухнастик	Собака	6	17
	18	Наті	Кролик	2	18

Рисунок 3.5. Заповнені дані Pets

Джерело: сформовано автором на основі виконаного дослідження

Yukie\LOCALDB#15...dbo.Appointments						
	AppointmentId	ClientId	PetId	VeterinarianId	ServiceId	Appointment...
	31	12	12	4	1	2025-06-16 07:0...
	32	12	12	1	5	2025-06-16 13:0...
	33	14	14	2	2	2025-06-16 08:0...
	34	15	15	5	1	2025-06-16 07:3...
	35	15	15	2	4	2025-06-16 08:3...
	36	16	16	3	5	2025-06-16 09:0...
	37	17	17	1	5	2025-06-16 07:0...
	38	17	17	3	2	2025-06-16 10:0...
	39	18	18	5	5	2025-06-16 09:3...
*	NULL	NULL	NULL	NULL	NULL	NULL

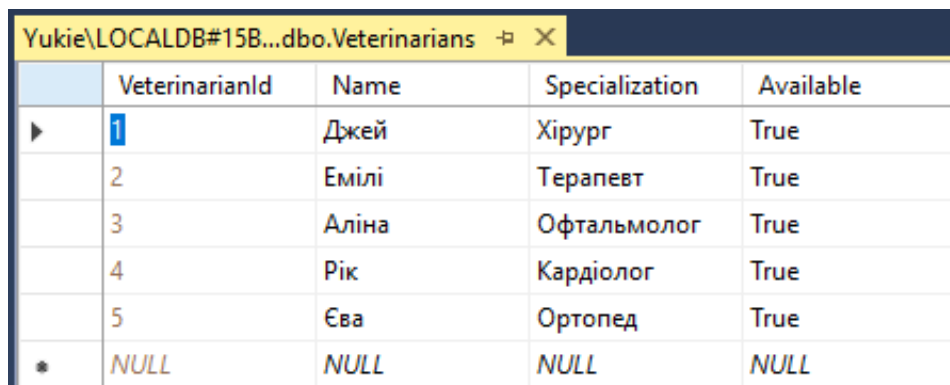
Рисунок 3.6. Заповнені дані Appointments

Джерело: сформовано автором на основі виконаного дослідження

Yukie\LOCALDB#15B...Db - dbo.Services				
	ServiceId	Name	Price	Currency
▶	1	Загальний огляд	500,00	UAH
	2	Вакцинація	750,00	UAH
	3	Видалення кліщів	300,00	UAH
	4	УЗД діагностика	1200,00	UAH
	5	Хірургічне втручання	2500,00	UAH
*	NULL	NULL	NULL	NULL

Рисунок 3.7. Заповнені дані Services

Джерело: сформовано автором на основі виконаного дослідження



	VeterinarianId	Name	Specialization	Available
▶	1	Джей	Хірург	True
	2	Емілі	Терапевт	True
	3	Аліна	Офтальмолог	True
	4	Рік	Кардіолог	True
	5	Єва	Ортопед	True
*	NULL	NULL	NULL	NULL

Рисунок 3.8. Заповнені дані Veterinarians

Джерело: сформовано автором на основі виконаного дослідження

3.2 Технічне забезпечення

3.2.1 Загальні положення та схема автоматизації

ІС електронного запису та управління чергою для ветеринарної клініки розробляється на базі архітектури клієнт-сервер. Серверна частина системи, ядро логіки, база даних будуть розгорнуті на віддаленому сервері в дата центрі. Користувачі системи(клієнт, адміністратор) матимуть доступ до функціоналу через вебінтерфейс, з будь-якого пристрою підключеного до Інтернету. Адміністраторам будуть надані робочі місця в клініці.

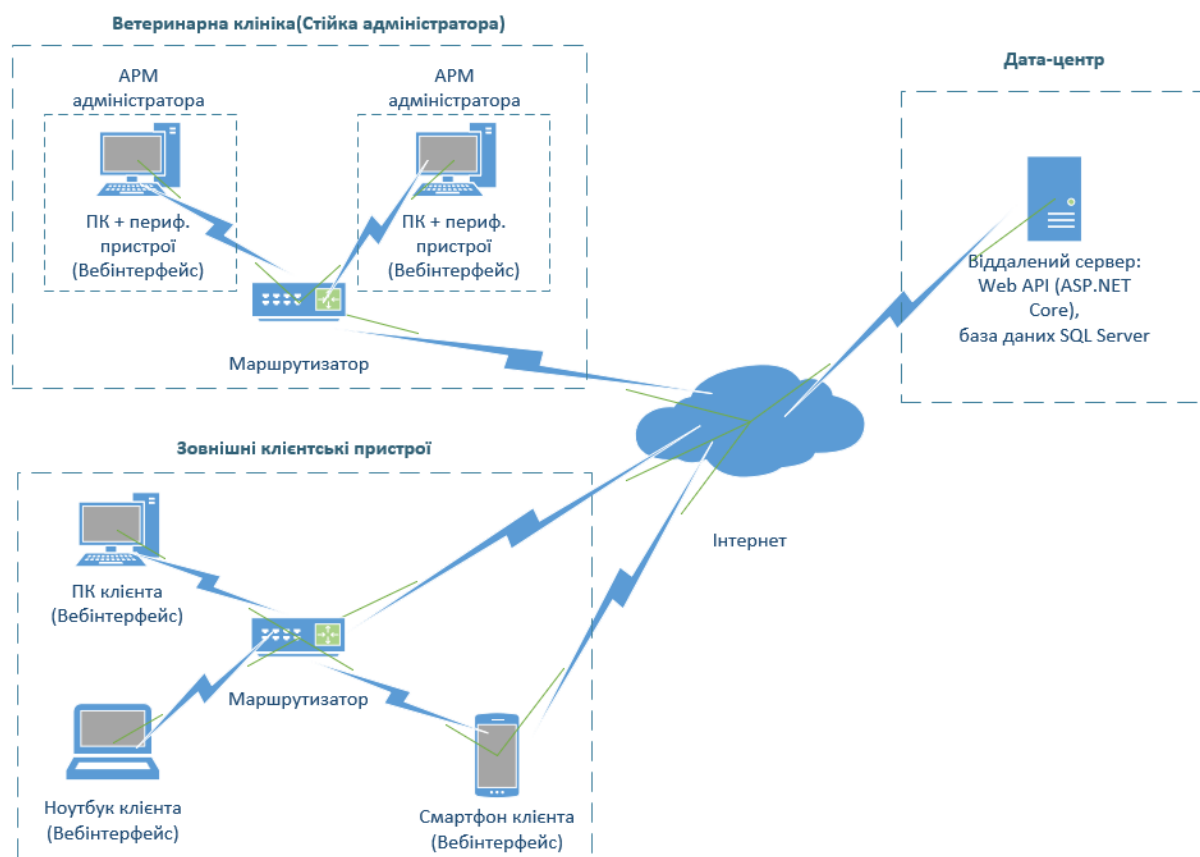


Рисунок 3.9 – Загальна схема автоматизації

Джерело: сформовано автором на основі виконаного дослідження

3.2.2 Структура комплексу технічних засобів

Система електронного запису та управління чергою виконана за архітектурою клієнт-сервер з використанням вебінтерфейсу. Її структура обрана з урахуванням вимог надійності, зменшенні витрат на адміністрування та зручності користування для клієнтів і адміністраторів клініки. Функції/процеси надані через інтерфейс. Тобто можна взаємодіяти з системою на будь-якому пристрої якщо є Інтернет з використанням протоколу HTTPS.

Наявні відповідності нефункціональним вимогам до мінімальних затримок, надійності та сумісності з браузером. Серверна частина системи розміщена на віддаленому фізичному сервері. Вона функціонує на сервері з параметрами: 6 ядра (x64), 16 ГБ оперативної пам'яті, SSD диск 1 ТБ обсяг, встановлена ОС Windows. Має безперебійну роботу системи, автоматичне резервне копіювання й захист через HTTPS.

База даних SQL Server, а бекенд ASP.NET Core. Вона не вимагає інтеграції з іншими автоматизованими системами, але через архітектуру можна розширити її інтерфейсами API. У клініці мінімальна кількість АРМ лише 2 для адміністраторів, які управляють чергою. Доступ клієнтів до системи проводиться персональними смартфонами, ноутбуками, ПК через вебінтерфейс. Додаткове апаратне забезпечення не використовується.

Розміщення технічних засобів на об'єктах керування не вимагає специфічних умов. Робочі місця адміністраторів розташовані у приміщеннях, згідно загальним вимогам техніки безпеки, мають електромережу, вентиляцію й освітлення.

Захист технічних засобів передбачає дотримання фізичної безпеки доступу до робочих ПК, використання паролів до облікових записів, а ще використання HTTPS для шифрування даних передавання між клієнтами та сервером. Сервер в дата центрі захищений засобами контролю доступу, джерелами живлення, засобами пожежогасіння та захистом від недозволеного доступу.

Вибір типу ПК і серверів обрано враховуючи потенційне навантаження.

Оскільки система не використовує ні датчики, ні маніпулятори чи пускові пристрої, ні технічні засоби в обчислювальній мережі об'єкта, має персональні ПК і маршрутизатори підключення до Інтернету.

Із периферійних пристроїв існують стандартні засоби введення/виведення інформації: монітори, миші й клавіатури, система не вимагає використання додаткових засобів.

Функціонуванням засобів займається мінімальна кількість персоналу. Користувачі (адміністратори, клієнти) працюють з системою з власних або наданих їм ПК, смартфонів або ноутбуків. Знаючи що серверна частина на стороні хостингу, обслуговування фізичного обладнання покладається на спеціалістів дата центру.

3.2.3 Опис автоматизованого робочого місця

Задумано використання одного типу АРМ - АРМ адміністраторів. Робочих місць є двоє для взаємодії з клієнтами через вебінтерфейс. Вони можуть переглядати та управляти чергою.

Щоб гарантувати стабільну та коректну роботу АРМ адміністратора передбачені мінімальні системні вимоги: процесор Intel Core i5 6 ядр й частота мінімум 2.9 ГГц, оперативна пам'ять 16 ГБ і вище, SSD об'ємом від 512 ГБ, ОС Windows 10-11.

З периферійних пристроїв АРМ адміністратора має: монітор з діагоналлю від 24 дюймів Full HD, клавіатура, миша. Периферійні пристрої вибрані для легкості використання щодня.

Кожне АРМ підключено до мережі Інтернет за допомогою дротового з'єднання Ethernet через маршрутизатор клініки швидкість не менше 100 мега біт в секунду. Коли настає тимчасове відключення основного каналу є резервне підключення через Wi-Fi або мобільний Інтернет. Заради убезпечення інформаційного обміну застосовується HTTPS.

Немає нагальної потреби, але якщо зміниться рівень загроз або з'являться нові безпекові вимоги, можна буде реалізувати апаратне шифрування або підключення через VPN.

3.2.4 Схема мережі передачі даних

Загальна структура мережі передачі даних має 3 логічні окремі смислові сегменти: локальну мережу ветеринарної клініки, клієнтську мережу та віддалену серверну частину в дата-центрі. Всі об'єднані мережею Інтернет, відповідають за взаємодію користувачів із сервером ІС. Схема наведена нижче:

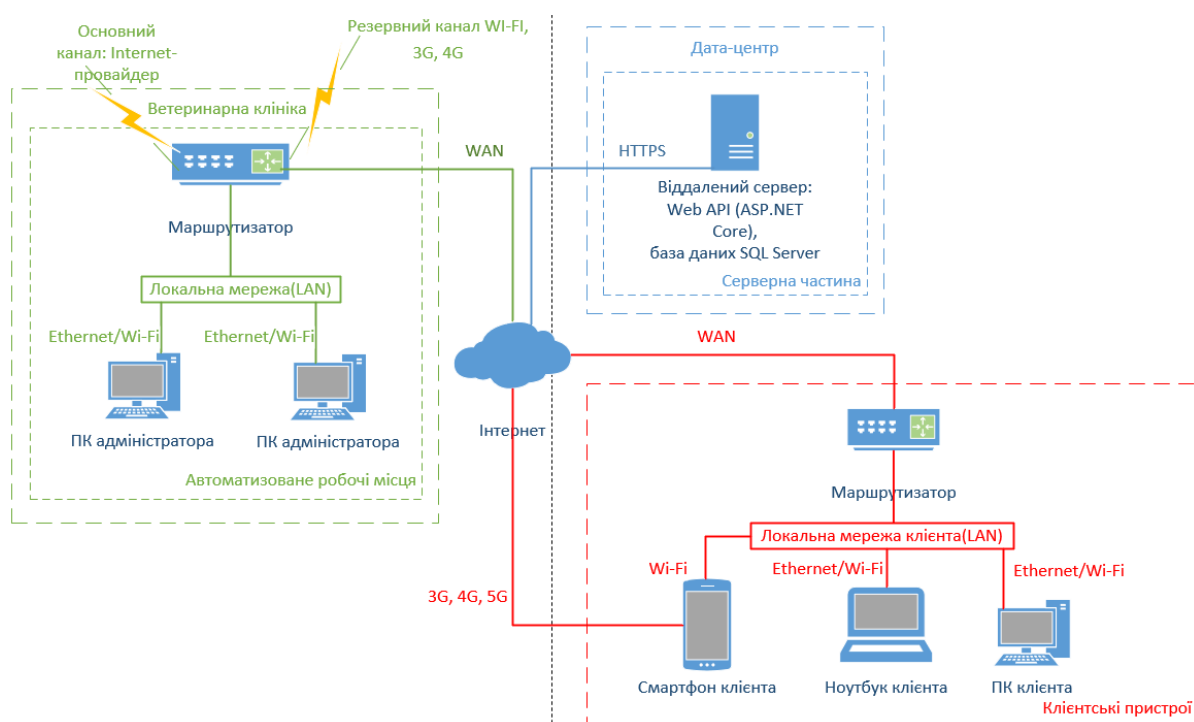


Рисунок 3.10 – Схема мережі передачі даних

Джерело: сформовано автором на основі виконаного дослідження

Клініка містить локальну мережу LAN з'єднує два автоматизовані робочих місць адміністраторів із маршрутизатором з кабельним або бездротовим з'єднанням.

Маршрутизатор клініки має підключення до основного каналу зв'язку від провайдера з характеристиками Ethernet, 100 Мбіт/с, static IP та резервного каналу через Wi-Fi, 3G, 4G.

Резервний варіант потрібен для доступу до сервера в разі відмови основного каналу.

Всього в клініці існує 1 маршрутизатор, 2 АРМ кожен по одному мережевому кабелю UTP-8 та є бездротове резервне підключення.

У клієнтській частині присутні наступні пристрої користувачів: смартфони, ноутбуки та персональні комп'ютери. Вони підключаються до локального маршрутизатора за допомогою Ethernet, Wi-Fi або 3G, 4G, 5G.

Між клієнтом і сервером дані передаються Інтернетом по захищеному HTTPS порту 443.

Частина сервера представлена як віддалений фізичний сервер в дата центрі з реалізованим Web API та базою даних SQL.

Сервер підтримує безперебійну роботу ІС, до нього звертаються як адміністратори клініки, так і клієнти. Сервер орієнтований на 1000 одночасно активних клієнтів, має канал зв'язку зі швидкістю від 100 Мбіт за секунду й HTTPS підключення до сервера.

Між всіма пристроями для з'єднання використано стандартні засоби передачі даних: Ethernet кабелі типу UTP-8, Wi-Fi у якості допоміжного каналу. У ролі маршрутизаторів рекомендовано використовувати моделі типу дводіапазонні гігабітні Wi-Fi маршрутизатори так, як вони підтримують 2.4/5 гіга герци й мають порти Gigabit.

Вимоги каналів зв'язку: основний канал клініки 100 Мбіт/с із статичною IP-адресою, мобільне або домашнє з'єднання клієнтам на швидкості більше 20 Мбіт за секунду.

Типове навантаження абонентів. Адміністратори ветеринарної клініки працюють з ІС 8 годин на добу 5 днів на тиждень управляючи і переглядаючи чергу. Власники улюбленців посилають запити, дивляться історію та залишають онлайн записи на прийом, тобто десь 10 хвилин активності на одного. Очікуване споживання каналу від одного користувача не перевищує 1 Мбіт в секунду.

Мережеве обладнання з середнім гарантійним терміном 36 місяців із можливістю розширення. Періодичність огляду маршрутизаторів одноразово

кожен рік. Усі компоненти повинні мати захист від нестабільної напруги і відповідати стандарту ЕМІ. Якщо виникла поломка окремі елементи можуть бути замінені.

3.3 Програмне забезпечення

3.3.1 Структура програмного забезпечення

Наступний за порядком фрагмент бакалаврської роботи присвячений на структуру програмного забезпечення. Кожна структура має компоненти, що її складають.

Інформаційної системи структура розділена 3 різні компоненти, проте всі вони є важливими в наданні повноцінної роботи функціонування системи й експлуатації її.

Перший компонент - системне ПЗ. Загалом воно установлює саме базове середовище роботи всієї системи, містить як операційні системи, утиліти, драйвери, так і середовища виконання на пару з інструментальними засобами, вони вже необхідні для маніпуляції інформації та без коливальної роботи техніки.

Далі прикладне або спеціалізоване програмне забезпечення, є другим компонентом ПЗ і знаючи термін виступає для реалізації прикладних задач, охоплює вебінтерфейси, частину серверу з API, модулі логіки з засобами роботи з БД, як SQL Server.

Остання частина ПЗ називається програмною документацією. Документація створена в засобі Enterprise Architect і включає технічні описи, інструкції, сценарії використання, формуляри, контрольні приклади та коди системи.

Розподілене ПЗ на 3 частини, утворено комплексний устрій для розроблення, впровадження та підтримки застосунку. Різні блоки зображено на схемі наведеній відразу після цього речення з поділом на частини ПЗ.

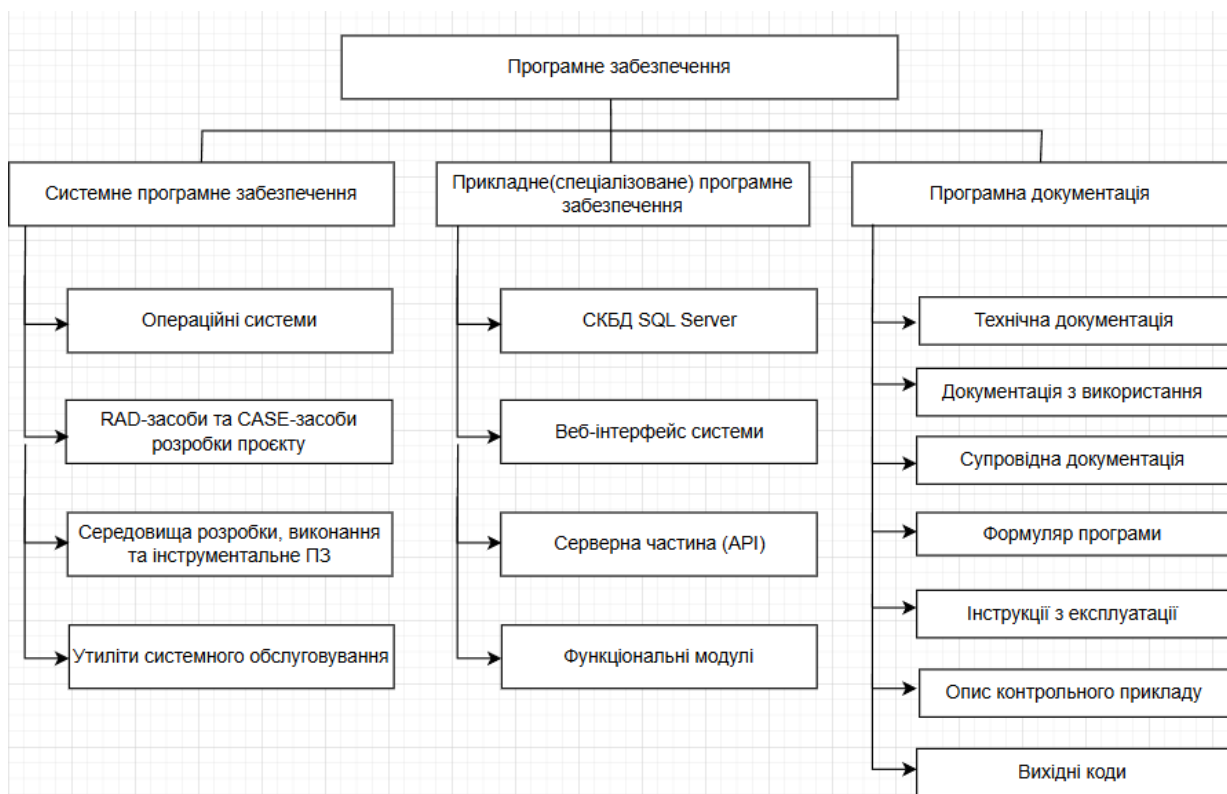


Рисунок 3.11 – Високорівнева схема програмного забезпечення

Джерело: сформовано автором на основі виконаного дослідження

3.3.2 Системне програмне забезпечення

Основою роботи всіх інформаційних систем виступає системне ПЗ.

Системне програмне забезпечення - це програмне забезпечення, яке забезпечує платформу для іншого програмного забезпечення[29].

Система ветклініки реалізована на основі стеку ASP.NET Core Web API, Blazor Web Assembly і бази SQL. Стек забезпечення був обраний через різні вимоги до розширення в майбутньому та підтримки сучасних технологій веброзробки.

Microsoft Windows 10/11 один із компонентів системного ПЗ, операційна система. Сумісна з .NET та з SQL Server. Вона потрібна створити середовище для розгортки серверного блоку. Ще одна інтеграція із .NET база даних Microsoft SQL Server, вона треба для розробки та обслуговування.

Далі CASE-засоби. Необхідно писати проєктну документацію і створювати багато всього як діаграми або моделі баз даних. Звичайно по роботі можна побачити, що вибір для всього цього пав на Enterprise Architect. Архітектура інфраструктури, підготовлена з урахуванням цілей безпеки для компаній, є вирішальною основою надійності систем і даних[30].

Варто зазначити також і RAD засіб Visual Studio, у якому розробляється і пишеться код системи. Visual Studio включає в себе компілятори, інструменти для завершення коду, графічні дизайнери[31]. Можна тестувати систему і розгорнути будь-які застосунки з сказаним вище стеком ASP.NET Core й Blazor.

Список системного ПЗ не закінчується. Наступними існують сервісні утиліти підтримки безпеки, обслуговування системи із засобами резервного копіювання.

Інший утиліт інструментальні засоби. Мається на увазі компілятори, інтерпретатори, тобто необхідні компоненти процесу створення та модифікації додатку. Інфраструктурою отримано стабільну роботу ІС ветеринарної клініки. Початком якої є етап розробки і потім її подальша експлуатація.

3.3.3 Прикладне програмне забезпечення

Другий стовпчик схеми програмного забезпечення становить прикладне.

Система повинна мати в собі елементи для створення бізнес логіки, елемент який відповідає за взаємодію з клієнтами. Інструменти прикладного ПЗ створюють цілісне середовище орієнтоване на вирішення конкретних завдань.

В ІС для ветеринарної клініки основним блоком можна виділити саме вебсистему. Щоб вона працювала послуговувались компоненти ASP.NET Web API для серверної частини та Blazor Web Assembly. Перший до серверної, другий для клієнтської частини. Звичайно можна сказати те, як така архітектура дійсно доволі практично розділяє процеси між інтерфейсом і логікою в системі з підтримкою масштабованості її.

Потім йде прикладне ПЗ модулі обробки даних. Обробляються дані про тварин, управління записами до всіх ветеринарів. Через що адміністратору потрібний окремий інтерфейс з підтримкою управління записами, не лише видалення, а й динамічне зміни часу або ветеринара. До речі, його інтерфейс корисний для перегляду статистики системи. Тим часом, клієнтам розроблено вебінтерфейс окремий, вони можуть в ньому реєструватися, записуватися на прийом, і після створення записів переглянути інформацію про запис.

Продовжуємо, наступна серверна частина, яка виконує роль посередника. Вона є зв'язком користувача і бази даних. ASP.NET Web API реалізує логіку системи через REST API.

Як і говорилося раніше в описі про ОС, СКБД SQL Server дуже гарно працює з .NET середовищем, як наслідок для роботи системи дані зберігаються і БД має високу продуктивність.

3.3.4 Програмна документація

Остання програмна документація. Процес створення програмного забезпечення складається з кількох видів діяльності та фаз[32].

Документація є обов'язковою частиною програмного забезпечення для всіх систем, особливо для ІС ветеринарної клініки. Розписано її правильне впровадження, як використовувати і зберегти працездатність.

Відомості створені з відповідністю до сучасних вимог і підготовлені засобом проєктування Enterprise Architect. Документація містить формуляр, з зазначенням: характеристик системи, складових компонентів, відомостей про правила експлуатації.

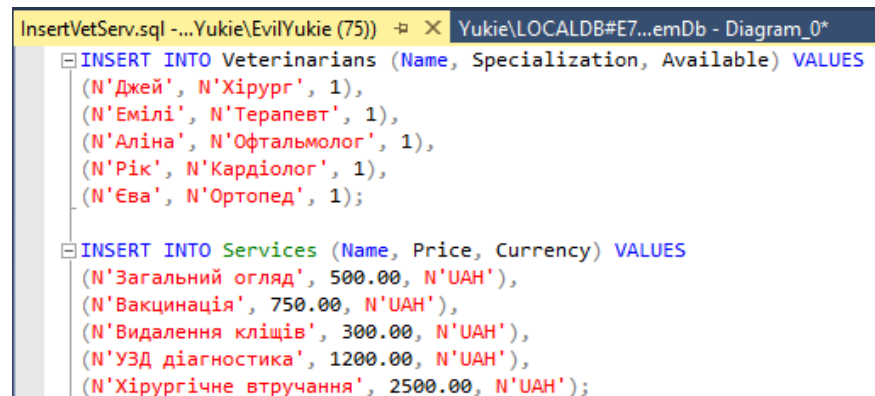
Крім того, важливо те, що подано і призначення програмного комплексу з сферами його застосування, сфери ведення обліку, наприклад пацієнтів , тобто улюбленців, адміністрування записів на прийом до закладу.

Потім ще є документація про мінімальні технічні вимоги до обладнання стабільної роботи. Туди входить рекомендована ОС, обсяг потрібної пам'яті, місце на диску та налаштування мережі.

Варто виділити увагу керівництву системного програміста, там в ньому детально пишеться порядок різних налаштувань, розгортань, резервних копіювань й обслуговувань системи відповідно до умов роботи.

Ще один опис контрольного прикладу роботи. Він існує як гарант того, що все працює і з ІС все нормально. Можна перевірити коректність встановлення та готові розроблені базові функції. Приклад тут, це ініціалізація даних у Service, Veterinarians таблицях БД VetclinicSystemDb.

З допомогою INSERT запиту додано тестові дані в Veterinarians й Service, наповнені по 5 різноманітних записів.



```

InsertVetServ.sql - ...Yukie\EvilYukie (75)  X Yukie\LOCALDB#E7...emDb - Diagram_0*
INSERT INTO Veterinarians (Name, Specialization, Available) VALUES
(N'Джей', N'Хірург', 1),
(N'Емілі', N'Терапевт', 1),
(N'Аліна', N'Офтальмолог', 1),
(N'Рік', N'Кардіолог', 1),
(N'Єва', N'Ортопед', 1);

INSERT INTO Services (Name, Price, Currency) VALUES
(N'Загальний огляд', 500.00, N'UAH'),
(N'Вакцинація', 750.00, N'UAH'),
(N'Видалення кліщів', 300.00, N'UAH'),
(N'УЗД діагностика', 1200.00, N'UAH'),
(N'Хірургічне втручання', 2500.00, N'UAH');

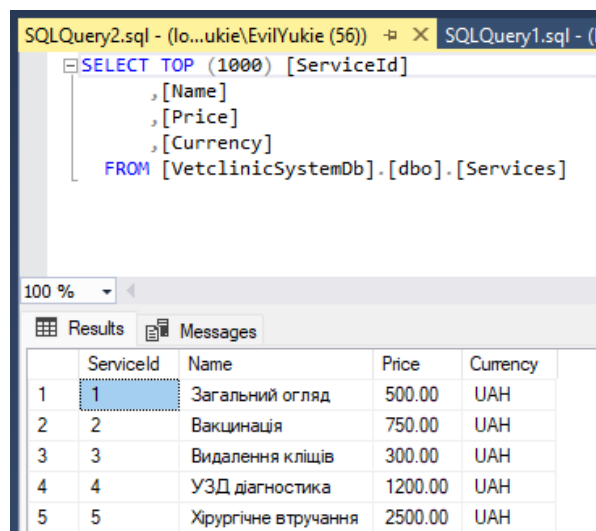
```

Рисунок 3.12 – Команда додавання

Джерело: сформовано автором на основі виконаного дослідження

Після у таблицях з'явилися дані про ветеринарів й послуги.

Першу заповнено іменами, спеціалізацією та доступністю на запис, другу назвами послуг клініки, ціною й її валютою.



```

SQLQuery2.sql - (lo...ukie\EvilYukie (56))  X SQLQuery1.sql - (
SELECT TOP (1000) [ServiceId]
, [Name]
, [Price]
, [Currency]
FROM [VetclinicSystemDb].[dbo].[Services]

```

	ServiceId	Name	Price	Currency
1	1	Загальний огляд	500.00	UAH
2	2	Вакцинація	750.00	UAH
3	3	Видалення кліщів	300.00	UAH
4	4	УЗД діагностика	1200.00	UAH
5	5	Хірургічне втручання	2500.00	UAH

Рисунок 3.13 – Результат у Service

Джерело: сформовано автором на основі виконаного дослідження

The screenshot shows a SQL query window with the following query:

```
SELECT TOP (1000) [VeterinarianId]
, [Name]
, [Specialization]
, [Available]
FROM [VetclinicSystemDb].[dbo].[Veterinarians]
```

Below the query, the 'Results' tab is active, displaying a table with 5 rows and 5 columns: VeterinarianId, Name, Specialization, and Available. The data is as follows:

	VeterinarianId	Name	Specialization	Available
1	1	Джей	Хірург	1
2	2	Емілі	Терапевт	1
3	3	Аліна	Офтальмолог	1
4	4	Рік	Кардіолог	1
5	5	Єва	Ортопед	1

Рисунок 3.14 – Результат у Veterinarians

Джерело: сформовано автором на основі виконаного дослідження

Всі поля заповнені. Контрольне тестування зроблено аби підвередити роботу БД і не перейматися далі розробляючи функціонал ПЗ.

Не може бути тільки опис прикладу ще повинні бути наведені і тексти коду. Їх відповідно можна продивитися в Додатку В.

3.4 Результати реалізації інформаційної системи

Реалізовано ІС електронного запису та управління чергою для ветеринарної клініки. Основна реалізація стосувалась створенню функціональності клієнтам системи. Побудовано робочий варіант з врахуванням процесів визначених на початку проєктування, вони зв'язані з записом і підтримують необхідну взаємодію між клієнтом і системою.

Розроблена варіант інформаційної системи успішно має функції: реєстрація нового клієнта, авторизація клієнта, створення запису на прийом до ветеринара, перегляд історії записів, на які він записався в системі.

Усі коди проєкту наведені в Додатку В.

Реєстрація користувача реалізована через відповідну форму з обов'язковим заповненням і валідацією полів. Наведено інтерфейс форми реєстрації:

Регістрація

Email:

Пароль:

Підтвердіть пароль:

Ваше ім'я:

Кличка тварини:

Вид тварини:

Вік тварини (років):

[Зареєструватися](#)

[Вже є обліковий запис? Увійти](#)

[Повернутися на головну сторінку клієнта](#)

Рисунок 3.15 – Форма реєстрації клієнта

Джерело: сформовано автором на основі виконаного дослідження

Авторизація побудована на сучасному механізмі автентифікації з використанням токенів JWT, що гарантує безпеку обміну даними та захист особистої інформації користувача. Готова форма авторизації:

Вхід

Email:

Пароль:

[Увійти](#)

[Немає облікового запису? Зареєструватися](#)

[Повернутися на головну сторінку клієнта](#)

Рисунок 3.16 – Форма авторизації

Джерело: сформовано автором на основі виконаного дослідження

Сервіс запису на прийом забезпечує вибір дати, години, послуги та ветеринара, враховуючи доступність вільного часу, що дозволяє уникнути конфліктів у розкладі. Форма запису:

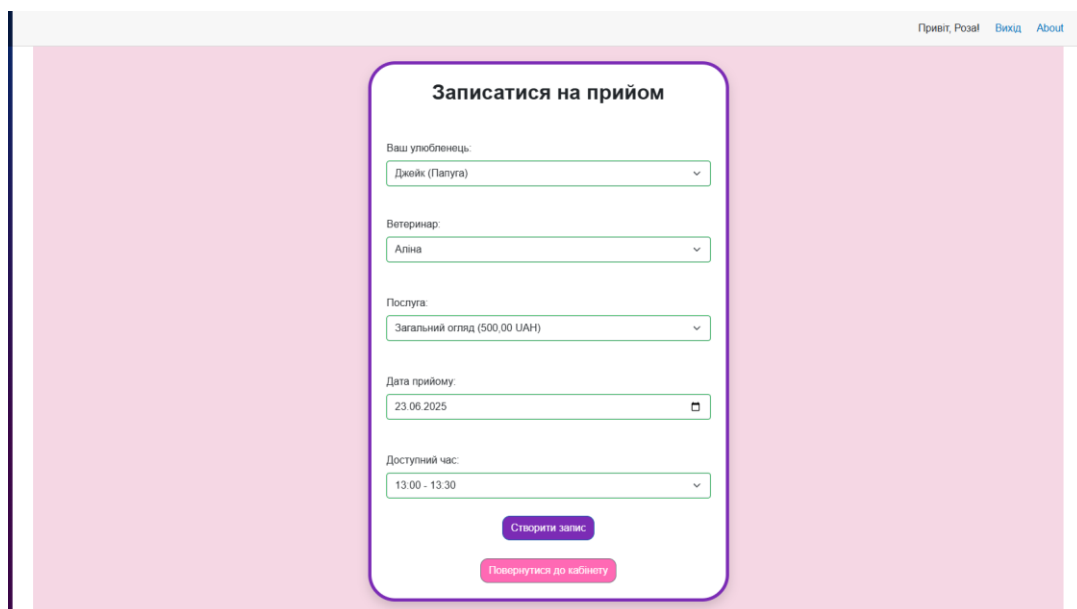
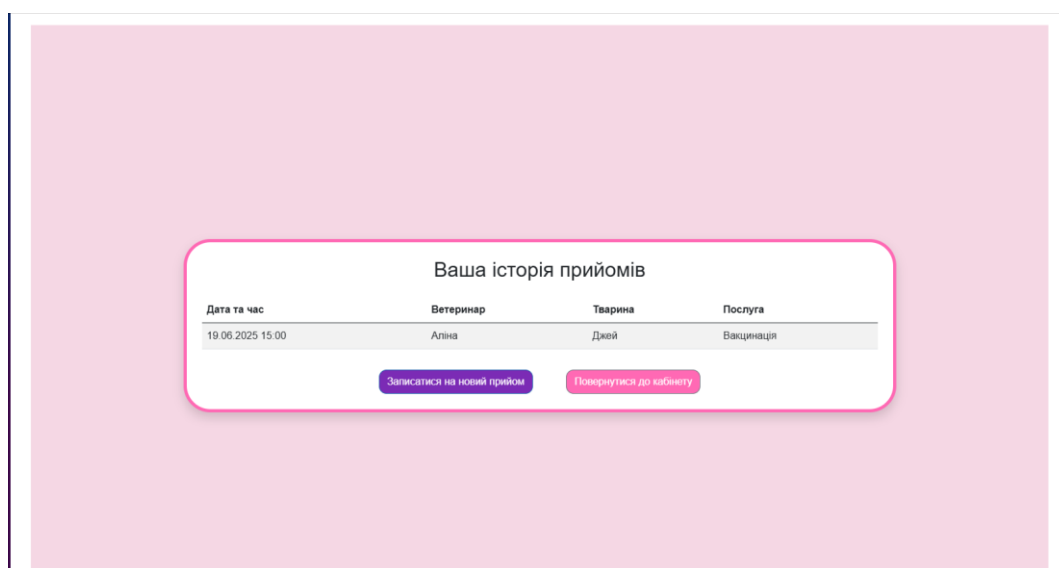


Рисунок 3.17 – Форма запису на прийом

Джерело: сформовано автором на основі виконаного дослідження

Перегляд записів надає клієнту зручну таблицю з історією усіх звернень, їхнім статусом і коротким описом. Наступний рисунок містить сторінку з історією створених записів:



Дата та час	Ветеринар	Тварина	Послуга
19.06.2025 15:00	Аліна	Джек	Вакцинація

Рисунок 3.18 – Сторінка перегляду історії

Джерело: сформовано автором на основі виконаного дослідження

На жаль через обмеження обсягу бакалаврської роботи й часових ресурсів, не вдалося реалізувати функціональні вимоги і для адміністратора. Немає функцій таких, як авторизація адміністратора, зміна порядку записів у черзі, переадресація запису до іншої черги, видалення запису з черги, а ще відображення записів у чергах. Хоча названий функціонал був гарно спроектований діаграмами, моделями БД, логікою, реалізація залишається до подальшого кроку розроблення. Так, як архітектура системи містить урахування розширюваності, помилок при інтегруванні адміністративних функцій не буде.

Перевірено роботу ПЗ в контрольному прикладі. Утворено тестові акаунти клієнтів, проведено реєстрацію, вхід, створено записи на прийом і перевірено відображення історії. Тестові дії проходили через інтерактивний інтерфейс системи, дані збережені в SQL Server, інтегрованою з Entity Framework Core. Всі дії обробляються правильно, функції працюють як сказано в вимогах.

Результати роботи підтвердили робочий стан системи в межах функціоналу клієнта. Всі функції клієнта працюють, розроблене рішення готове до використання в житті.

Інформаційна система може бути основою для подальшого розширення, вони можуть містити додавання адміністративної частини, аналітики, сповіщень, інтеграцію з оплатою тощо. Система демонструє успішну будову й відповідність вимогам.

ВИСНОВКИ

Результатом розроблення, виконаної в межах бакалаврської дипломної роботи, є досягнення мети, тобто розробити інформаційну систему електронного запису та управління чергою у ветеринарній клініці для підвищення рівня обслуговування клієнтів, оптимізації навантаження робітників й надання легкої комунікації між адміністрацією й власниками домашніх тварин. Вдалось здійснити повноцінне проєктування, моделювання та реалізацію функціональної частини клієнта системи.

Розроблена система вирішує актуальну проблему організації взаємодії між клієнтами ветеринарної клініки та адміністраторами. Вона отримала простий комфортний інтерфейс здійснення реєстрації, авторизації клієнта, оформлення запису на прийом, перегляду історії. Завдяки чому система спрощує процес обслуговування, зменшує навантаження на реєстратуру, процеси перенесені в електронний формат і може виконуватись безпосередньо клієнтами через власний кабінет.

Ведення електронної історії візитів формує базу даних з усією необхідною інформацією про улюбленців, тому клієнти можуть отримувати доступ до повної інформації прийому перед його початком, аналізувати попередні звернення, звіряти дату і час запису, у разі потреби. Ще база може використовуватись для аналітики, звітності або моніторингу популярності запису до певного ветеринара чи знаходити час пік роботи закладу.

ІС розроблена зважаючи до існуючих потреб ветеринарних клінік, може легко адаптуватися до різних умов роботи в маленьких приватних практик з простотою і швидкістю впровадження, і ще в середніх закладах, які потребують масштабованого рішення з можливістю майбутнього розширення разом з іншими системами.

Через задіяння вебтехнологій система може працювати в локальному і в хмарному середовищі, має захист даних. Під час реалізації проєкту задіяно методи системного аналізу, моделювання та інженерії вимог. Збудовано діаграми варіантів використання, діаграми послідовності, діаграми класів, вимог згідно стандарту SysML ISO/IEC 19514:2017. На додачу примінено об'єктно-орієнтовані принципи, тому проєктування має розширювану архітектуру. В основі технічної реалізації покладені новітні вебтехнології:

- ASP.NET Core Web API, основа серверної логіки,
- Blazor WebAssembly для побудови інтерактивного односторінкового клієнтського інтерфейсу,
- SQL Server як система керування базами даних.

Реалізована архітектура client-server з підтримкою запитів API і механізмом авторизації через токени JWT.

Аналізовано аналогічні існуючі програмні реалізації на ринку. Потім ще проведено оцінку функціональних можливостей, роботу і комфорт інтерфейсу, перспективи інтегрування. Результати аналізу демонструють і допомагають визначити обґрунтовані переваги створеної системи, тобто можливість її розширення, відкритість інтеграції з іншими зовнішніми сервісами, гарний дизайн і те, як система відповідає архітектурою актуальним принципам розроблення ПЗ. Реалізовані функції системи з боку клієнта пройшли внутрішнє тестування і показали коректність їхньої роботи у прикладі роботи.

Запропоновані механізми електронного запису, обліку історії тощо, несуть в собі покращення стандарту обслуговування клієнтів і полегшення роботи адміністраторів. Система має потенціал впровадження у реальні ветеринарні клініки для автоматизації обліку клієнтів.

Іншим моментом, який треба відзначити виступає апробація результатів на студентській науково-практичній конференції. Проаналізовано те, які технології зараз задіюють в системах ветеринарних клінік. Розібрано як часто

використовують ІКТ-рішення, проблеми наявні зараз і варіанти покращення систем.

Особистий внесок здобувача полягає у повному циклі розроблення проєкту: від написання вимог, будови моделей, великої кількості діаграм і сценаріїв, до програмної реалізації та перевірки результатів роботи.

Самостійно реалізовано серверну логіку, клієнтський інтерфейс, структуру бази даних. На початку проєктування сформульовано вимоги через обмеження і потребу зараз, побудовано діаграми, розроблено документацію системи.

Результати дипломної роботи мають теоретичну і практичну цінність. Проєкт створено на базі сучасних технологій: ASP.NET Core Web API, Blazor WebAssembly, SQL Server та Entity Framework Core.

У майбутньому можна включити адміністративний функціонал, який був вже спроектований, додати аналітичний модуль, повідомлення, інші системи, які збігаються з темою автоматизації процесів електронного запису і управління чергою. Розроблена система може стати основою до подібних рішень у галузі ветеринарної медицини.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Учасники проєктів Вікімедіа. Ветеринарна медицина– Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Ухвалення_рішень (дата звернення: 11.06.2025).
2. Ветеринарія. Panpolski. URL: <https://panpolski.com.ua/uk/tehnikumy/veterynariya/> (дата звернення: 11.06.2025).
3. Beazley H. What is Veterinary Software? | MarketBox. MarketBox | Appointment Scheduling & Route Optimization Software. URL: <https://www.gomarketbox.com/blog/what-is-veterinary-software> (date of access: 11.06.2025).
4. Vet clinic management system: Types & Best Software to Know. Bigscal - Software Development Company. URL: <https://www.bigscal.com/blogs/healthcare-industry/vet-clinic-management-system/> (date of access: 11.06.2025).
5. Types of Positions at a Veterinary Practice. Highland Veterinary Clinic | Evansville IN Vet. URL: <https://www.highlandvet.net/types-of-positions-at-a-veterinary-practice> (date of access: 11.06.2025).
6. Учасники проєктів Вікімедіа. Ухвалення рішень – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Ухвалення_рішень (дата звернення: 11.06.2025).
7. Управлінське рішення, його роль, функції, види, типології. URL: <https://studies.in.ua/polit-men-shpora/1691-36-41-upravlnske-rshennya-yogo-rol-funkcyi-vidi-tipologyi.html> (дата звернення: 11.06.2025).
8. CVA M. D. D. The veterinary practice operating system. DVM 360. URL: <https://www.dvm360.com/view/the-veterinary-practice-operating-system> (date of access: 11.06.2025).
9. David P. A. Path dependence: A foundational concept for historical social science //Law, economics and evolutionary theory. – Edward Elgar Publishing, 2011.

10. External and Internal Data. External and Internal Data Differences in the Data Marketplace. URL: <https://revelate.co/blog/external-and-internal-data-differences-in-the-data-marketplace/> (date of access: 11.06.2025).
11. What Are The Different Types Of IT Solutions?. DynamiQes. URL: <https://dynamiqes.com/types-of-it-solutions/> (date of access: 11.06.2025).
12. Veterinary Practice Management Software - NaVetor. NaVetor. URL: <https://www.navetor.com/> (date of access: 11.06.2025).
13. Contributors to Wikimedia projects. Input–output model - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Input–output_model (date of access: 11.06.2025).
14. Frontiers | The potential application of artificial intelligence in veterinary clinical practice and biomedical research / O. C. Akinsulie et al. Frontiers. URL: <https://www.frontiersin.org/journals/veterinary-science/articles/10.3389/fvets.2024.1347550/full> (date of access: 11.06.2025).
15. What is a Workstation? - Intel. Intel. URL: <https://www.intel.com/content/www/us/en/products/docs/systems-devices/workstations/what-is-a-workstation>. (date of access: 11.06.2025).
16. Hippo Manager Overview. Hippo Manager General Information. URL: <https://pitchbook.com/profiles/company/104062-06> (date of access: 11.06.2025).
17. Hippo Manager. Softwareadvice Hippo Manager Benefits, Features & Pricing. URL: <https://www.softwareadvice.com/veterinary/hippo-manager-profile/>.
18. Covetrus Pulse EVetPractice Software: Reviewing Features, Pricing – Spotsaas Blog. spotSaaS Blog. URL: <https://www.spotsaas.com/blog/evetpractice-veterinary-practice-solution/> (date of access: 11.06.2025).
19. Veterinary software support. Evetpractice veterinary software support. URL: <https://www.iag-usa.com/support/veterinary-software-support/evetpractice-veterinary-software-support/> (date of access: 11.06.2025).

20. NaVetor - CareCredit Tech Partner Directory. Health and Wellness - CareCredit. URL: <https://www.carecredit.com/providers/partnerships/directory/navetor/> (date of access: 11.06.2025).
21. NaVetor - CompareYourTech. NaVetor URL: <https://compareyourtech.com/listing/navetor/> (date of access: 11.06.2025).
22. IEC/IEEE 29148:2018 Systems and software engineering - Life cycle processes - Requirements engineering
23. Enterprise Architect Blog. Enterprise Architect Blog – SparxSystems Central Europe. URL: https://blog.sparxsystems.de/en_GB/ (date of access: 11.06.2025).
24. Understanding Business Requirements for Project Success. awork: Work management for agencies. URL: <https://www.awork.com/glossary/business-requirements> (date of access: 11.06.2025).
25. Output. Examples of output on a computer. URL: <https://www.computerhope.com/jargon/o/output.htm> (date of access: 11.06.2025).
26. Avallone G, Rasotto R, Chambers JK, et al. Review of Histological Grading Systems in Veterinary Medicine. Veterinary Pathology. 2021;58(5):809-828. doi:10.1177/0300985821999831
27. Scenario Builder User Guide. Scenario Builder | Enterprise Architect User Guide. URL: https://sparxsystems.com/enterprise_architect_user_guide/15.2/tools_ba_scenario_builder (date of access: 11.06.2025).
28. Contributors to Wikimedia projects. erwin Data Modeler - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Erwin_Data_Modeler (date of access: 11.06.2025).
29. GeeksforGeeks. What is System Software? – Definition, Types, Examples and More - GeeksforGeeks. GeeksforGeeks. URL: <https://www.geeksforgeeks.org/system-software/> (date of access: 11.06.2025).

30. Judijanto L. et al. Edge of enterprise architecture in addressing cyber security threats and business risks //International Journal Software Engineering and Computer Science (IJSECS). – 2023. – Т. 3. – №. 3. – С. 386-396.
 31. Visual Studio: IDE and Code Editor for Software Developers and Teams. Visual Studio. URL: <https://visualstudio.microsoft.com/#vs-section> (date of access: 11.06.2025).
 32. CERN Document Server. URL: <https://cds.cern.ch/record/383260/files/p135.pdf> (дата звернення: 11.06.2025).
- Sundarapandian, V. (2009). "7. Queueing Theory". Probability, Statistics and Queueing Theory. PHI Learning. ISBN 978-81-203-3844-9.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МІНІСТЕРСТВО З ПИТАНЬ СТРАТЕГІЧНИХ ГАЛУЗЕЙ ПРОМІСЛОВОСТІ УКРАЇНИ
Київський національний економічний університет імені Вадима Гетьмана
Вроцлавський університет економіки та бізнесу (Польща)
Інститут інновацій та конкуренції Макса Планка (Німеччина, Мюнхен)
Національний університет Ірландії в Мейнутаї (Ірландія, Мейнут)
Інститут кібернетики імені В.М. Глушкова НАН України
Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України
Інститут проблем штучного інтелекту МОН та НАН України
Інститут телекомунікацій і глобального інформаційного простору НАН України
Запорізький національний університет
Івано-Франківський національний технічний університет нафти і газу
Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Національний університет «Острозька академія»
Одеський національний економічний університет
Харківський національний університет радіоелектроніки
Хмельницький національний університет

СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ В УПРАВЛІННІ

Збірник матеріалів

VI Міжнародної науково-практичної конференції
молодих вчених, аспірантів і студентів

10–11 квітня 2025 р.



Рисунок А.1 – Перша сторінка збірника тез

<i>Яценко В.К.</i> ПЕРСПЕКТИВИ АВТОМАТИЗАЦІЇ СИСТЕМ УПРАВЛІННЯ З ВИКОРИСТАННЯМ ІІІ	270
<i>Shevchuk I.M.</i> DESIGNING A SEISMIC DATA SYSTEM: VISUALIZATION AND CLUSTERING	272
ПЛАТФОРМА	
ЦИФРОВА ЕКОНОМІКА. ІНДУСТРІЯ 4.0 ТА 5.0. SMART – ЕКОНОМІКА. ІНТЕГРОВАНІ ІНТЕЛЕКТУАЛЬНІ ВИРОБНИЧІ СИСТЕМИ	274
<i>Алексеев М. А.</i> РОЗВИТОК ЦИФРОВОЇ ЛОГІСТИКИ В УМОВАХ ІНДУСТРІЇ 5.0	274
<i>Варич К. А., Гострик О. М.,</i> ДОПОВНЕНА ТА ВІРТУАЛЬНА РЕАЛЬНІСТЬ У ВИРОБНИЧИХ ПРОЦЕСАХ ІНДУСТРІЇ 4.0	275
<i>Гальчич І. П.</i> ФОРМУВАННЯ МЕДІАЦІЙНИХ НАВИЧОК МАЙБУТНІХ СОЦІАЛЬНИХ ПРАЦІВНИКІВ У ПРОЦЕСІ ПРАКТИЧНОЇ ПІДГОТОВКИ ЗАСОБАМИ ЦИФРОВИХ ТЕХНОЛОГІЙ	277
<i>Герасименко Д. О.</i> SMART-МЕДИЦИНА В КОНТЕКСТІ ЦИФРОВОЇ ЕКОНОМІКИ: ВИКЛИКИ ТА ПЕРСПЕКТИВИ ДЛЯ ПРИВАТНОГО СЕКТОРУ	280
<i>Гуріна К.П.</i> ЦИФРОВА ТРАНСФОРМАЦІЯ ЯК ОСНОВНИЙ ФАКТОР ВІДНОВЛЕННЯ ЕКОНОМІКИ ТА СТАЛОГО РОЗВИТКУ УКРАЇНИ	282
<i>Карпіна І.С.</i> АНАЛІЗ ІСНУЮЧИХ ІНФОРМАЦІЙНИХ СИСТЕМ ДЛЯ УПРАВЛІННЯ ОХОРОННИМИ ПРОЦЕСАМИ ТА ЇХ ЕФЕКТИВНІСТЬ	284
<i>Кишук А. В.</i> ЦИФРОВА ТРАНСФОРМАЦІЯ В СФЕРІ ВЕТЕРИНАРНИХ ПОСЛУГ: РОЗРОБЛЕННЯ ІНТЕГРОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ЧЕРГОЮ ТА ЕЛЕКТРОННОГО ЗАПИСУ	285
<i>Крикунова А.К., Скіцько В.І.</i> ПЕРЕХІД ВІД ІНДУСТРІЇ 4.0 ДО ІНДУСТРІЇ 5.0 НА ПРИКЛАДІ «SIEMENS»	287
<i>Кубрицька М. М., Скіцько В.І.</i> ГАРМОНІЯ ЛЮДИНИ І МАШИНИ – НОВА ПАРАДИГМА INDUSTRY 5.0.....	289
<i>Луцук М.С.</i> ЗАРУБІЖНИЙ ДОСВІД ВПРОВАДЖЕННЯ ЦИФРОВИХ ТЕХНОЛОГІЙ У ГОТЕЛЯХ	290
<i>Савінков Н.Д., Волкова Н.В.</i> ІНТЕРНЕТ РЕЧЕЙ У МОНІТОРИНГУ ЕКОЛОГІЧНИХ ЗАГРОЗ.....	292

Рисунок А.2 – Зміст збірника тез

ЦИФРОВА ТРАНСФОРМАЦІЯ В СФЕРІ ВЕТЕРИНАРНИХ ПОСЛУГ: РОЗРОБЛЕННЯ ІНТЕГРОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ЧЕРГОЮ ТА ЕЛЕКТРОННОГО ЗАПИСУ

Нині всі сфери життя потребують впровадження цифрової трансформації, включаючи ветеринарну медицину. Традиційні методи запису тварин на прийом не є досконалими через збільшення кількості пацієнтів та постійно зростаючі вимоги до якості обслуговування. Більшість

285

клієнтів хочуть мати швидкий і надійний сервіс, що стимулює впровадження інноваційних цифрових рішень у цю сферу. З'явилася необхідність у розробці такої інтегрованої системи управління чергою та електронного запису, яка допоможе не лише оптимізувати роботу, а й покращити комунікацію клієнтів з закладом.

Розглянемо проблеми наявні зараз у ветеринарних клініках, які потребують використання прогресивних інформаційних технологій. Основними недоліками наразі можна виокремити обмеження звичайних систем обліку, слабка інтеграція з сучасними ІКТ-рішеннями, висока завантаженість працівників і питання безпеки даних. Інтеграція блокчейн-технологій і мобільних додатків з використанням передових методів машинного навчання пропонують рішення для автоматизації процесів і, разом з тим, здатні надати надійний захист інформації та адаптивність системи до мінливих ринкових умов.

Поточних труднощів в організації запису та управління чергою у ветеринарних клініках можна запобігти, розробивши інтегровану вебсистему, яка застосовує базис сучасних програмних платформ з високою продуктивністю. Як варіант, наприклад, використання передових JS-фреймворків – React, Angular, Vue.js, і побудова бекенд-частини із застосуванням C# та ASP.NET Core дає змогу оперативно обробляти запити із чіткою синхронізацією з базою даних клініки. Крім того, інтеграція новітніх блокчейн-технологій вирішує проблеми, пов'язані з конфіденційною інформацією і дає змогу підвищити довіру користувачів. Є можливість досягти прозорості операцій, створивши розподілений та незмінний реєстр даних з автоматизацією транзакцій, впроваджуючи смарт-контракти. По-справжньому інноваційно вплинуло на цифрову трансформацію використання машинного навчання у сфері ветеринарних послуг, що сприяє більш досконалому плануванню роботи персоналу. Проблема прогнозу завантаженості роботи клініки й раціонального використання ресурсів вирішується використанням алгоритмів ШІ, інтегруючи аналітичні модулі в систему для аналізу даних про записи, відвідування та формування черг. Оперативний доступ власників до інформації і сповіщення про зміни у розкладі можливий завдяки інтеграції синхронізованого з вебсистемою мобільного додатку, що усуває труднощі в комунікації між медичним закладом і пацієнтами. Такий комплексний підхід до розробки системи не лише автоматизує адміністративні процеси, а й створює гнучке середовище для цифрової трансформації ветеринарних послуг.

Розроблення інтегрованої системи управління чергою та електронного запису відкриває нові можливості для модернізації роботи і є важливим кроком для підвищення якості обслуговування та оптимізації внутрішніх процесів клініки. Інноваційні технології, такі як блокчейн, машинне навчання та мобільні додатки, дають змогу не лише скоротити час очікування пацієнтів, а й забезпечити високий рівень безпеки та прозорості обробки даних, що є ключовим аспектом сучасної цифровізації.

Список використаних джерел

1. Beyer K. Customer experiences drive the need of innovation in veterinary practices // *Procedia Computer Science*. – 2023. – Т. 225. – С. 3094-3103.
2. Karolina B. Unlocking the potential of ICT innovation in veterinary healthcare: The pathway to improve practices and business model // *Procedia Computer Science*. – 2023. – Т. 225. – С. 4775-4784.
3. Iqbal N. et al. A novel blockchain-based integrity and reliable veterinary clinic information management system using predictive analytics for provisioning of quality health services // *Ieee Access*. – 2021. – Т. 9. – С. 8069-8098.

Науковий керівник: Мозгалі О. П., д.е.н., професор.

Рисунок А.3 – Тези автора дослідження

Опис складових бази даних в термінах DDL

```
CREATE TABLE Users (  
  UserId INT IDENTITY(1,1) PRIMARY KEY,  
  Name NVARCHAR(100) NOT NULL,  
  PasswordHash NVARCHAR(255) NOT NULL,  
  Email NVARCHAR(255) UNIQUE NOT NULL,  
  Role NVARCHAR(20) NOT NULL CHECK (Role IN (N'Клієнт', N'Адміністратор'))  
);  
  
CREATE TABLE Pets (  
  PetId INT IDENTITY(1,1) PRIMARY KEY,  
  OwnerId INT NOT NULL,  
  Name NVARCHAR(100) NOT NULL,  
  Species NVARCHAR(100) NOT NULL,  
  AgeYears INT,  
  FOREIGN KEY (OwnerId) REFERENCES Users(UserId) ON DELETE CASCADE  
);  
  
CREATE TABLE Veterinarians (  
  VeterinarianId INT IDENTITY(1,1) PRIMARY KEY,  
  Name NVARCHAR(100) NOT NULL,  
  Specialization NVARCHAR(100),  
  Available BIT DEFAULT 1  
);  
  
CREATE TABLE Services (  
  ServiceId INT IDENTITY(1,1) PRIMARY KEY,  
  Name NVARCHAR(100) NOT NULL,  
  Price DECIMAL(10,2) NOT NULL,  
  Currency NVARCHAR(10)  
);  
  
CREATE TABLE Appointments (  
  AppointmentId INT IDENTITY(1,1) PRIMARY KEY,  
  ClientId INT NOT NULL,  
  PetId INT NOT NULL,  
  VeterinarianId INT NOT NULL,  
  ServiceId INT NOT NULL,
```

```
AppointmentDateTime DATETIME2 NOT NULL,  
FOREIGN KEY (ClientId) REFERENCES Users(UserId) ON DELETE NO ACTION,  
FOREIGN KEY (PetId) REFERENCES Pets(PetId) ON DELETE CASCADE,  
FOREIGN KEY (VeterinarianId) REFERENCES Veterinarians(VeterinarianId) ON DELETE CASCADE,  
FOREIGN KEY (ServiceId) REFERENCES Services(ServiceId) ON DELETE CASCADE  
);  
  
CREATE TABLE Queues (  
QueueId INT IDENTITY(1,1) PRIMARY KEY,  
VeterinarianId INT NOT NULL,  
AppointmentId INT NOT NULL,  
QueuePosition INT NOT NULL,  
Status NVARCHAR(100),  
FOREIGN KEY (VeterinarianId) REFERENCES Veterinarians(VeterinarianId) ON DELETE CASCADE,  
FOREIGN KEY (AppointmentId) REFERENCES Appointments(AppointmentId) ON DELETE CASCADE,  
CONSTRAINT UQ_Queues_VeterinarianId_QueuePosition UNIQUE (VeterinarianId, QueuePosition)
```

Коди програми VetclinicSys.Client:

Program.cs

```
using Microsoft.AspNetCore.Components.Web;
using Microsoft.AspNetCore.Components.WebAssembly.Hosting;
using VetclinicSys.Client.Services;
using VetclinicSys.Client.Handlers;
using Blazored.LocalStorage;
using Microsoft.AspNetCore.Components.Authorization;
using VetclinicSys.Client.Auth;

namespace VetclinicSys.Client
{
    public class Program
    {
        public static async Task Main(string[] args)
        {
            var builder = WebAssemblyHostBuilder.CreateDefault(args);
            builder.RootComponents.Add<App>("#app");
            builder.RootComponents.Add<HeadOutlet>("head::after");

            builder.Services.AddBlazoredLocalStorage();

            builder.Services.AddTransient<CustomAuthorizationHandler>();

            builder.Services.AddScoped<AuthenticationStateProvider, CustomAuthStateProvider>();

            builder.Services.AddScoped<CustomAuthStateProvider>();

            builder.Services.AddAuthorizationCore();

            builder.Services.AddHttpClient("ServerApi", client =>
            {
                client.BaseAddress = new Uri(builder.HostEnvironment.BaseAddress);
            })
                .AddHttpMessageHandler<CustomAuthorizationHandler>();

            builder.Services.AddScoped<IAuthService>(sp =>
            {
                var httpClientFactory = sp.GetRequiredService<IHttpClientFactory>();
                var httpClient = httpClientFactory.CreateClient("ServerApi");
                var localStorage = sp.GetRequiredService<ILocalStorageService>();

                var authStateProvider = sp.GetRequiredService<CustomAuthStateProvider>();
                return new AuthService(httpClient, localStorage, authStateProvider);
            });
            builder.Services.AddScoped<IQueueService>(sp =>
                new QueueService(sp.GetRequiredService<IHttpClientFactory>().CreateClient("ServerApi")));

            builder.Services.AddScoped<IAppointmentService>(sp =>
                new
                AppointmentService(sp.GetRequiredService<IHttpClientFactory>().CreateClient("ServerApi")));

            builder.Services.AddScoped<IUserService>(sp =>
                new UserService(sp.GetRequiredService<IHttpClientFactory>().CreateClient("ServerApi")));

            builder.Services.AddScoped<IVeterinarianService>(sp =>
                new
                VeterinarianService(sp.GetRequiredService<IHttpClientFactory>().CreateClient("ServerApi")));

            builder.Services.AddScoped<IServiceService>(sp =>
                new
                ServiceService(sp.GetRequiredService<IHttpClientFactory>().CreateClient("ServerApi")));

            await builder.Build().RunAsync();
        }
    }
}
```

RoleSelection.razor

```
@page "/"
@inject NavigationManager NavigationManager
<PageTitle>Вибір ролі</PageTitle>

<div class="d-flex flex-column align-items-center justify-content-center vh-100"
style="background-color: #F8C8DC;">
    <h1 class="mb-4">Оберіть вашу роль</h1>
    <div class="d-grid gap-3 col-6 mx-auto">
        <button class="btn btn-primary btn-lg" style="background-color: #6A0DAD; color:
white;" @onclick="NavigateToClientHome">Клієнт</button>
        <button class="btn btn-primary btn-lg" style="background-color: #6A0DAD; color:
white;" @onclick="NavigateToAdminHome">Адміністратор</button>
    </div>
</div>
@code {
    private void NavigateToClientHome()
    {
        NavigationManager.NavigateTo("/client");
    }

    private void NavigateToAdminHome()
    {
        NavigationManager.NavigateTo("/admin");
    }
}
```

ClientHome.razor

```
@page "/client"
@inject NavigationManager NavigationManager
<PageTitle>Клієнт</PageTitle>

<div class="d-flex flex-column align-items-center justify-content-center vh-100"
style="background-color: #F8C8DC;">
    <h1 class="mb-4">Вітаємо, Клієнте!</h1>
    <div class="d-grid gap-3 col-6 mx-auto">
        <button class="btn btn-success btn-lg" style="background-color: #6A0DAD; color:
white;" @onclick="NavigateToRegister">Зареєструватися</button>
        <button class="btn btn-primary btn-lg" style="background-color: #6A0DAD; color:
white;" @onclick="NavigateToLogin">Увійти</button>
        <button class="btn btn-link mt-3" @onclick="NavigateToRoleSelection">Повернутися
до вибору ролі</button>
    </div>
</div>
@code {
    private void NavigateToRegister()
    {
        NavigationManager.NavigateTo("/client/register");
    }

    private void NavigateToLogin()
    {
        NavigationManager.NavigateTo("/client/login");
    }

    private void NavigateToRoleSelection()
    {
        NavigationManager.NavigateTo("/");
    }
}
```

ClientRegistration.razor

```

@page "/client/register"
@using VetclinicSys.Client.Services
@using VetclinicSys.Shared.DTOS
@inject IAuthService AuthService
@inject NavigationManager NavigationManager

<PageTitle>Реєстрація Клієнта</PageTitle>

<div class="d-flex flex-column align-items-center justify-content-center vh-100"
style="background-color: #F8C8DC;">
    <div class="card p-4 shadow" style="width: 25rem;">
        <h2 class="card-title text-center mb-4">Реєстрація Клієнта</h2>
        <EditForm Model="@registrationDto" OnValidSubmit="@HandleRegistration">
            <DataAnnotationsValidator />
            <ValidationSummary />

            <div class="mb-3">
                <label for="email" class="form-label">Email:</label>
                <InputText id="email" class="form-control" @bind-
Value="registrationDto.Email" />
            </div>

            <div class="mb-3">
                <label for="password" class="form-label">Пароль:</label>
                <InputText id="password" type="password" class="form-control" @bind-
Value="registrationDto.Password" />
            </div>

            <div class="mb-3">
                <label for="confirmPassword" class="form-label">Підтвердіть
пароль:</label>
                <InputText id="confirmPassword" type="password" class="form-control"
@bind-Value="registrationDto.ConfirmPassword" />
            </div>

            <div class="mb-3">
                <label for="name" class="form-label">Ваше ім'я:</label>
                <InputText id="name" class="form-control" @bind-
Value="registrationDto.Name" />
            </div>

            <div class="mb-3">
                <label for="petName" class="form-label">Кличка тварини:</label>
                <InputText id="petName" class="form-control" @bind-
Value="registrationDto.PetName" />
            </div>

            <div class="mb-3">
                <label for="species" class="form-label">Вид тварини:</label>
                <InputText id="species" class="form-control" @bind-
Value="registrationDto.Species" />
            </div>

            <div class="mb-3">
                <label for="ageYears" class="form-label">Вік тварини (років):</label>
                <InputNumber id="ageYears" class="form-control" @bind-
Value="registrationDto.AgeYears" />
            </div>

            <div class="mb-3">
                <div class="alert alert-danger">@errorMessage</div>
            </div>

            <div class="mb-3">
                <div class="alert alert-success">@registrationSuccess</div>
            </div>
        </EditForm>
    </div>
</div>

```

```

    {
        <div class="alert alert-success">Реєстрація успішна! Тепер ви можете <a
href="/client/login">увійти</a>.</div>
    }

    <button type="submit" class="btn btn-success w-100" style="background-color:
#6A0DAD; color: white;" disabled="@isLoading">
        @if (isLoading)
        {
            <span class="spinner-border spinner-border-sm" role="status" aria-
hidden="true"></span>
            <span>Завантаження...</span>
        }
        else
        {
            <span>Зареєструватися</span>
        }
    </button>
</EditForm>
<div class="text-center mt-3">
    <p>Вже є обліковий запис? <a href="/client/login">Увійти</a></p>
    <button class="btn btn-link" @onclick="NavigateToClientHome">Повернутися на
головну сторінку клієнта</button>
</div>
</div>
</div>

```

```

@code {
    private UserRegisterDto registrationDto = new UserRegisterDto();
    private string? errorMessage;
    private bool isLoading = false;
    private bool registrationSuccess = false;

    private async Task HandleRegistration()
    {
        isLoading = true;
        errorMessage = null;
        registrationSuccess = false;

        try
        {
            if (registrationDto.Password != registrationDto.ConfirmPassword)
            {
                errorMessage = "Паролі не співпадають.";
                isLoading = false;
                return;
            }

            var response = await AuthService.Register(registrationDto);

            if (response.IsSuccess)
            {
                registrationSuccess = true;
                registrationDto = new UserRegisterDto();
            }
            else
            {
                errorMessage = response.Message ?? "Реєстрація не вдалася. Перевірте
дані.";
            }
        }
        catch (ApplicationException ex)
        {
            errorMessage = ex.Message;
        }
    }
}

```

```

        catch (Exception)
        {
            errorMessage = "Виникла невідома помилка при реєстрації. Спробуйте пізніше.";
        }
        finally
        {
            isLoading = false;
        }
    }

    private void NavigateToClientHome()
    {
        NavigationManager.NavigateTo("/client");
    }
}

```

ClientLogin.razor

```

@page "/client/login"
@using VetclinicSys.Client.Services
@inject IAuthService AuthService
@inject NavigationManager NavigationManager
@inject AuthenticationStateProvider AuthStateProvider
@inject ILocalStorageService LocalStorage

<PageTitle>Вхід для Клієнта</PageTitle>
<div class="d-flex flex-column align-items-center justify-content-center vh-100"
style="background-color: #F8C8DC;">
    <div class="card p-4 shadow" style="width: 25rem;">
        <h2 class="card-title text-center mb-4">Вхід для Клієнта</h2>
        <EditForm Model="@loginDto" OnValidSubmit="@HandleLogin">
            <DataAnnotationsValidator />
            <ValidationSummary />

            <div class="mb-3">
                <label for="email" class="form-label">Email:</label>
                <InputText id="email" class="form-control" @bind-Value="loginDto.Email"
/>
            </div>

            <div class="mb-3">
                <label for="password" class="form-label">Пароль:</label>
                <InputText id="password" type="password" class="form-control" @bind-
Value="loginDto.Password" />
            </div>

            <if (!string.IsNullOrEmpty(errorMessage))>
                <div class="alert alert-danger">@errorMessage</div>
            </if>

            <button type="submit" class="btn btn-primary w-100" style="background-color:
#6A0DAD; color: white;" disabled="@isLoading">
                <if (isLoading)>
                    <span class="spinner-border spinner-border-sm" role="status" aria-
hidden="true"></span>
                    <span>Завантаження...</span>
                </if>
                <else>
                    <span>Увійти</span>
                </else>
            </button>

```

```

        </EditForm>
        <div class="text-center mt-3">
            <p>Немає облікового запису? <a
href="/client/register">Зареєструватися</a></p>
            <button class="btn btn-link" @onclick="NavigateToClientHome">Повернутися на
головну клієнта</button>
        </div>
    </div>
</div>
@code {
    private LoginDto loginDto = new LoginDto();
    private string? errorMessage;
    private bool isLoading = false;
    protected override async Task OnInitializedAsync()
    {
        var authState = await AuthStateProvider.GetAuthenticationStateAsync();
        if (authState.User.Identity?.IsAuthenticated == true &&
authState.User.IsInRole("Client"))
        {
            NavigationManager.NavigateTo("/client/dashboard");
        }
    }
    private async Task HandleLogin()
    {
        isLoading = true;
        errorMessage = null;
        try
        {
            var response = await AuthService.Login(loginDto);
            var authState = await AuthStateProvider.GetAuthenticationStateAsync();
            if (authState.User.Identity?.IsAuthenticated == true &&
authState.User.IsInRole("Client"))
            {
                NavigationManager.NavigateTo("/client/dashboard", forceLoad: true);
            }
            else
            {
                await AuthService.Logout();
                errorMessage = "Ви не є клієнтом. Будь ласка, увійдіть через
адміністраторську форму або зареєструйтесь як клієнт.";
            }
        }
        catch (ApplicationException ex)
        {
            errorMessage = ex.Message;
        }
        catch (Exception)
        {
            errorMessage = "Виникла невідома помилка при вході. Спробуйте пізніше.";
        }
        finally
        {
            isLoading = false;
        }
    }
    private void NavigateToClientHome()
    {
        NavigationManager.NavigateTo("/client");
    }
}

```

ClientDashboard.razor

```

@page "/client/dashboard"
@using VetclinicSys.Client.Services

```

```

@using VetclinicSys.Shared.DTOs
@using Microsoft.AspNetCore.Components.Authorization
@inject IUserService UserService
@inject NavigationManager NavigationManager
@inject IAuthService AuthService

<PageTitle>Мій Кабінет Клієнта</PageTitle>
<AuthorizeView Roles="Client">
    <Authorized>
        <div class="d-flex flex-column align-items-center justify-content-center vh-100"
style="background-color: #F8C8DC;">
            <div class="card p-4 shadow" style="width: 30rem;">
                <h2 class="card-title text-center mb-4"> @UserProfile?.Name</h2>

                @if (isLoading)
                {
                    <p>Завантаження даних профілю...</p>
                    <div class="spinner-border text-primary" role="status">
                        <span class="visually-hidden">Loading...</span>
                    </div>
                }
                else if (userProfile != null)
                {
                    <div class="mb-3">
                        <strong>Електронна пошта:</strong> @UserProfile.Email
                    </div>

                    <div class="mb-3">
                        <strong>Ім'я улюбленця:</strong> @UserProfile.PetName
                    </div>

                    <hr class="my-4"/>

                    <h4 class="text-center mb-3">Що далі?</h4>
                    <div class="d-grid gap-2">
                        <button class="btn btn-primary btn-lg" style="background-color:
#6A0DAD; color: white;" @onclick="NavigateToAppointmentBooking">Записатися на
прийом</button>
                        <button class="btn btn-primary btn-lg" style="background-color:
#6A0DAD; color: white;" @onclick="NavigateToAppointmentHistory">Переглянути історію
прийомів</button>
                    </div>
                }
                else
                {
                    <div class="alert alert-danger">Не вдалося завантажити профіль
користувача. Будь ласка, спробуйте <a href="/client/login" >Увійти знову</a>.</div>
                }

                <div class="text-center mt-4">
                    <button class="btn btn-danger" @onclick="Logout">Вийти</button>
                </div>
            </div>
        </Authorized>
        <NotAuthorized>
            <div class="alert alert-warning text-center mt-5">
                Ви не авторизовані для доступу до цієї сторінки. Будь ласка, <a
href="/client/login">увійдіть</a> як клієнт.
            </div>
        </NotAuthorized>
    </AuthorizeView>

@code {
    private UserDto? userProfile;

```

```

private bool isLoading = true;

[CascadingParameter]
private Task<AuthenticationState>? authenticationState { get; set; }

protected override async Task OnInitializedAsync()
{
    var authState = await authenticationState;
    if (authState?.User?.Identity?.IsAuthenticated == true &&
authState.User.IsInRole("Client"))
    {
        userProfile = await UserService.GetCurrentUserProfileAsync();
        isLoading = false;
    }
    else
    {
        NavigationManager.NavigateTo("/client/login");
    }
}
private void NavigateToAppointmentBooking()
{
    NavigationManager.NavigateTo("/appointments/create");
}
private void NavigateToAppointmentHistory()
{
    NavigationManager.NavigateTo("/appointments/history");
}
private async Task Logout()
{
    await AuthService.Logout();
    NavigationManager.NavigateTo("/client");
}
}

```

AppointmentHistory.razor

```

@page "/appointments/history"
@using VetclinicSys.Client.Services
@using VetclinicSys.Shared.DTOS
@using Microsoft.AspNetCore.Components.Authorization
@inject IAppointmentService AppointmentService
@inject NavigationManager NavigationManager

<PageTitle>Історія прийомів</PageTitle>

<AuthorizeView Roles="Client">
    <Authorized>
        <div class="d-flex flex-column align-items-center justify-content-center vh-100"
style="background-color: #F8C8DC;">
            <div class="card p-4 shadow" style="width: 70rem;">
                <h2 class="card-title text-center mb-4">Ваша історія прийомів</h2>

                @if (isLoading)
                {
                    <div class="text-center">
                        <div class="spinner-border text-primary" role="status" >
                            <span class="visually-hidden">Завантаження...</span>
                        </div>
                        <p>Завантаження ваших записів...</p>
                    </div>
                }
                else if (!string.IsNullOrEmpty(errorMessage))
                {
                    <div class="alert alert-danger">@errorMessage</div>
                }
            }
        </div>
    </Authorized>

```

```

else if (appointments == null || !appointments.Any())
{
    <div class="alert alert-info text-center">
        У вас поки немає записів в історії.
        <br />
        <button class="btn btn-primary mt-2" style="background-color:
#6A0DAD; color: white;" @onclick="NavigateToCreateAppointment">
            Записатися на прийом
        </button>
    </div>
}
else
{
    <table class="table table-striped table-hover table-responsive">
        <thead>
            <tr>
                <th>Дата та час</th>
                <th>Ветеринар</th>
                <th>Тварина</th>
                <th>Послуга</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var appt in appointments)
            {
                <tr>
                    <td>@appt.AppointmentDateTime.ToLocalTime().ToString("dd.MM.yyyy HH:mm")</td>
                    <td>@appt.VeterinarianName</td>
                    <td>@appt.PetName</td>
                    <td>@appt.ServiceName</td>
                </tr>
            }
        </tbody>
    </table>
}

<div class="text-center mt-3">
    <button class="btn btn-secondary" style="background-color: #6A0DAD;
color: white;" @onclick="NavigateToClientDashboard">Повернутися до кабінету</button>
    <button class="btn btn-primary ms-2" style="background-color:
#6A0DAD; color: white;" @onclick="NavigateToCreateAppointment">Записатися на новий
прийом</button>
</div>
</div>
</Authorized>
<NotAuthorized>
    <div class="alert alert-warning text-center mt-5">
        Ви не авторизовані для доступу до цієї сторінки. Будь ласка, <a
href="/client/login">увійдіть</a> як клієнт.
    </div>
</NotAuthorized>
</AuthorizeView>

@code {
    private List<AppointmentHistoryItemDto>? appointments;
    private bool isLoading = true;
    private string? errorMessage;

    [CascadingParameter]
    private Task<AuthenticationState>? authenticationState { get; set; }

    protected override async Task OnInitializedAsync()
    {

```

```

        var authState = await authenticationState;
        if (authState?.User?.Identity?.IsAuthenticated == true &&
authState.User.IsInRole("Client"))
        {
            await LoadAppointments();
        }
        else
        {
            isLoading = false;
        }
    }

    private async Task LoadAppointments()
    {
        isLoading = true;
        errorMessage = null;
        try
        {
            appointments = await AppointmentService.GetHistoryAsync();
        }
        catch (ApplicationException appEx)
        {
            errorMessage = $"Помилка завантаження історії прийомів: {appEx.Message}";
            Console.WriteLine($"Application Exception in history page: {appEx.Message}");
        }
        catch (HttpRequestException httpEx)
        {
            errorMessage = $"Помилка завантаження історії прийомів: {httpEx.Message}.
Перевірте з'єднання з сервером.";
            Console.WriteLine($"HTTP Request Error in history page: {httpEx.Message}");
        }
        catch (Exception ex)
        {
            errorMessage = $"Сталася невідома помилка при завантаженні історії прийомів:
{ex.Message}";
            Console.WriteLine($"General Error in history page: {ex.Message}");
        }
        finally
        {
            isLoading = false;
        }
    }

    private void NavigateToClientDashboard()
    {
        NavigationManager.NavigateTo("/client/dashboard");
    }

    private void NavigateToCreateAppointment()
    {
        NavigationManager.NavigateTo("/appointments/create");
    }

```

VetclinicSys.System:

Program.cs

```

using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System.Text;
using VetclinicSys.Server.Data;
using Microsoft.AspNetCore.Identity;

var builder = WebApplication.CreateBuilder(args);

```

```

builder.Services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

builder.Services.AddScoped<IPasswordHasher<VetclinicSys.Shared.Models.User>,
PasswordHasher<VetclinicSys.Shared.Models.User>>();

builder.Services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
    options.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        ValidateIssuerSigningKey = true,
        ValidIssuer = builder.Configuration["Jwt:Issuer"],
        ValidAudience = builder.Configuration["Jwt:Audience"],
        IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]))
    };
});

builder.Services.AddControllersWithViews();
builder.Services.AddRazorPages();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddLogging(configure => configure.AddConsole());

builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowSpecificOrigin",
        builder => builder.WithOrigins("https://localhost:7067", "http://localhost:5215")
            .AllowAnyMethod()
            .AllowAnyHeader()
            .AllowCredentials());
});

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseWebAssemblyDebugging();
    app.UseSwagger();
    app.UseSwaggerUI();
}
else
{
    app.UseExceptionHandler("/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseBlazorFrameworkFiles();
app.UseStaticFiles();

app.UseRouting();

app.UseCors("AllowSpecificOrigin");

```

```

app.UseAuthentication();
app.UseAuthorization();

app.MapRazorPages();
app.MapControllers();
app.MapFallbackToFile("index.html");

app.Run();

```

UserController

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Security.Claims;
using VetclinicSys.Server.Data;
using VetclinicSys.Shared.DTOs;

namespace VetclinicSys.Server.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    [Authorize]
    public class UsersController : ControllerBase
    {
        private readonly AppDbContext _context;

        public UsersController(AppDbContext context)
        {
            _context = context;
        }

        [HttpGet("me/profile")]
        public async Task<IActionResult> GetCurrentUserProfile()
        {
            var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier);
            if (userIdClaim == null || !int.TryParse(userIdClaim.Value, out int userId))
            {
                return Unauthorized("Користувач не авторизований або ID не знайдено.");
            }

            var user = await _context.Users
                .Include(u => u.Pets)
                .FirstOrDefaultAsync(u => u.UserId == userId);

            if (user == null)
            {
                return NotFound("Профіль користувача не знайдено.");
            }

            var userDto = new UserDto
            {
                Id = user.UserId,
                Name = user.Name,
                Email = user.Email,
                Role = user.Role,
                PetName = user.Pets.FirstOrDefault()?.Name ?? "N/A"
            };

            return Ok(userDto);
        }
    }
}

```

```

[HttpGet("me/pets")]
[Authorize(Roles = "Client")]
public async Task<IActionResult> GetMyPets()
{
    var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier);
    if (userIdClaim == null || !int.TryParse(userIdClaim.Value, out int userId))
    {
        return Unauthorized("Користувач не авторизований або ID не знайдено.");
    }

    var pets = await _context.Pets
                            .Where(p => p.Owner.UserId == userId)
                            .ToListAsync();

    if (!pets.Any())
    {
        return NotFound("Тварини для цього користувача не знайдено.");
    }

    return Ok(pets);
}
}

```

RegistrationController

```

using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using VetclinicSys.Server.Data;
using VetclinicSys.Shared.Models;
using Microsoft.EntityFrameworkCore;
using VetclinicSys.Shared.DTOs;

namespace VetclinicSys.Server.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class RegistrationController : ControllerBase
    {
        private readonly ApplicationDbContext _context;
        private readonly IPasswordHasher<User> _passwordHasher;

        public RegistrationController(ApplicationDbContext context, IPasswordHasher<User>
passwordHasher)
        {
            _context = context;
            _passwordHasher = passwordHasher;
        }

        [HttpPost("register")]
        public async Task<IActionResult> Register(UserRegisterDto dto)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(new RegisterResponseDto
                {
                    IsSuccess = false,
                    Message = "Некоректні дані реєстрації.",
                });
            }

            if (await _context.Users.AnyAsync(u => u.Email == dto.Email))
                return BadRequest(new RegisterResponseDto { IsSuccess = false, Message =
"Обліковий запис з такою електронною поштою вже існує!" });

```

```

        var user = new User
        {
            Email = dto.Email,
            Name = dto.Name,
            Role = "Client"
        };

        user.PasswordHash = _passwordHasher.HashPassword(user, dto.Password);
        _context.Users.Add(user);
        await _context.SaveChangesAsync();

        var pet = new Pet
        {
            Owner = user,
            Name = dto.PetName,
            Species = dto.Species,
            AgeYears = dto.AgeYears
        };
        _context.Pets.Add(pet);
        await _context.SaveChangesAsync();

        return Ok(new RegisterResponseDto { IsSuccess = true, Message = "Реєстрація
успішна!" });
    }
}

```

AuthController

```

using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
using VetclinicSys.Server.Data;
using VetclinicSys.Shared.DTOS;
using VetclinicSys.Shared.Models;

namespace VetclinicSys.Server.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class AuthController : ControllerBase
    {
        private readonly AppDbContext _context;
        private readonly IPasswordHasher<User> _passwordHasher;
        private readonly IConfiguration _config;
        private readonly ILogger<AuthController> _logger;

        public AuthController(AppDbContext context, IPasswordHasher<User> passwordHasher,
            IConfiguration config, ILogger<AuthController> logger)
        {
            _context = context;
            _passwordHasher = passwordHasher;
            _config = config;
            _logger = logger;
        }

        [HttpPost("login")]
        public async Task<IActionResult> Login(LoginDto dto)
        {
            Console.WriteLine("Auth Controller Login method HIT!");
        }
    }
}

```

```

_logger.LogInformation("Auth Controller Login method HIT via ILogger!");

if (!ModelState.IsValid)
{
    Console.WriteLine("Login: Model state is invalid.");
    return BadRequest(new LoginResponseDto
    {
        IsSuccess = false,
        Message = "Некоректні дані входу.",
        Errors = ModelState.Values.SelectMany(v => v.Errors).Select(e =>
e.ErrorMessage).ToList());
    });
}

var user = await _context.Users.SingleOrDefaultAsync(u => u.Email ==
dto.Email);
if (user == null)
{
    Console.WriteLine("Login: User not found.");
    return Unauthorized(new LoginResponseDto { IsSuccess = false, Message =
"Неправильна електронна пошта або пароль!" });
}

var result = _passwordHasher.VerifyHashedPassword(user, user.PasswordHash,
dto.Password);
if (result == PasswordVerificationResult.Failed)
{
    Console.WriteLine("Login: Password verification failed.");
    return Unauthorized(new LoginResponseDto { IsSuccess = false, Message =
"Неправильна електронна пошта або пароль!" });
}

var claims = new[]
{
    new Claim(ClaimTypes.NameIdentifier, user.UserId.ToString()),
    new Claim(ClaimTypes.Role, user.Role),
    new Claim(ClaimTypes.Name, user.Name)
};

var jwtKey = _config["Jwt:Key"];
var jwtIssuer = _config["Jwt:Issuer"];
var jwtAudience = _config["Jwt:Audience"];

_logger.LogInformation($"JWT Key: {jwtKey?.Substring(0,
Math.Min(jwtKey.Length, 10))}...");
_logger.LogInformation($"JWT Issuer: {jwtIssuer}");
_logger.LogInformation($"JWT Audience: {jwtAudience}");
_logger.LogInformation($"User Role from DB: {user.Role}");

if (string.IsNullOrEmpty(jwtKey) || string.IsNullOrEmpty(jwtIssuer) ||
string.IsNullOrEmpty(jwtAudience))
{
    _logger.LogError("JWT configuration is incomplete. Key, Issuer, or
Audience is missing.");
    return StatusCode(StatusCodes.Status500InternalServerError, "Серверна
помилка: конфігурація JWT неповна.");
}

try
{
    var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtKey));
    var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);
    var token = new JwtSecurityToken(
        issuer: jwtIssuer,
        audience: jwtAudience,

```

```

        claims: claims,
        expires: DateTime.UtcNow.AddHours(2),
        signingCredentials: creds
    );

    var tokenString = new JwtSecurityTokenHandler().WriteToken(token);
    _logger.LogInformation($"Generated JWT Token (first 50 chars):
{tokenString.Substring(0, Math.Min(tokenString.Length, 50))}...");
    _logger.LogInformation($"Token Length: {tokenString.Length}");

    var handler = new JwtSecurityTokenHandler();
    try
    {
        var parsedToken = handler.ReadJwtToken(tokenString);
        _logger.LogInformation($"Server successfully parsed generated JWT.
Claims: {string.Join(", ", parsedToken.Claims.Select(c => $"{c.Type}={c.Value}"))}");
    }
    catch (Exception ex)
    {
        _logger.LogError($"Server FAILED to parse generated JWT. Error:
{ex.Message}");
    }

    Console.WriteLine("Login method END.");
    return Ok(new LoginResponseDto { IsSuccess = true, Token = tokenString,
Role = user.Role, Message = "Вхід успішний!" });
}
catch (Exception ex)
{
    _logger.LogError($"Error generating JWT token: {ex.Message}");
    Console.WriteLine($"Error generating JWT token: {ex.Message}");
    return StatusCode(StatusCode.Status500InternalServerError, $"Внутрішня
помилка сервера при генерації токена: {ex.Message}");
}
}
}

```

AppointmentController

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Security.Claims;
using VetclinicSys.Server.Data;
using VetclinicSys.Shared.DTOs;
using VetclinicSys.Shared.Models;
namespace VetclinicSys.Server.Controllers
{
    [ApiController]

    [Route("api/[controller]")]

    [Authorize]

    public class AppointmentController : ControllerBase
    {

        private readonly ApplicationDbContext _context;

        private readonly TimeSpan WORK_START_TIME = new TimeSpan(6, 0, 0);
        private readonly TimeSpan WORK_END_TIME = new TimeSpan(15, 0, 0);
        private readonly int DEFAULT_SERVICE_DURATION_MINUTES = 30;
    }
}

```

```

public AppointmentController(AppDbContext context)
{
    _context = context;
}

[HttpPost("create")]
public async Task<IActionResult> CreateAppointment(CreateAppointmentDto dto)
{
    var userIdString = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;
    if (string.IsNullOrEmpty(userIdString) || !int.TryParse(userIdString, out int
clientId))
    {
        return Unauthorized("Не вдалося визначити ідентифікатор клієнта.");
    }

    var pet = await _context.Pets.SingleOrDefaultAsync(p => p.PetId == dto.PetId
&& p.Owner.UserId == clientId);

    if (pet == null)
    {
        return BadRequest("Вибрана тварина не належить цьому клієнту або не
існує.");
    }

    var client = await _context.Users.FindAsync(clientId);
    var veterinarian = await
_context.Veterinarians.FindAsync(dto.VeterinarianId);
    var service = await _context.Services.FindAsync(dto.ServiceId);

    if (client == null || veterinarian == null || service == null)
    {
        return BadRequest("Недійсні дані клієнта, ветеринара або послуги.");
    }

    if (dto.AppointmentDateTime <= DateTime.UtcNow)
    {
        return BadRequest("Дата і час прийому не можуть бути в минулому.");
    }

    if (dto.AppointmentDateTime.DayOfWeek == DayOfWeek.Saturday ||
dto.AppointmentDateTime.DayOfWeek == DayOfWeek.Sunday)
    {
        return BadRequest("Запис неможливий у вихідні дні.");
    }
}

```

```

        var proposedStartTime = dto.AppointmentDateTime;

        var proposedEndTime =
            proposedStartTime.AddMinutes(DEFAULT_SERVICE_DURATION_MINUTES);

        if (proposedStartTime.TimeOfDay < WORK_START_TIME ||
            proposedEndTime.TimeOfDay > WORK_END_TIME)
        {
            return BadRequest($"Обраний час виходить за межі робочого графіку
            ({WORK_START_TIME:hh\\:mm} - {WORK_END_TIME:hh\\:mm}).");
        }

        var conflictingAppointment = await _context.Appointments
            .Where(a => a.Veterinarian.VeterinarianId == veterinarian.VeterinarianId &&
                a.AppointmentDateTime < proposedEndTime &&
                a.AppointmentDateTime.AddMinutes(DEFAULT_SERVICE_DURATION_MINUTES) > proposedStartTime)
            .AnyAsync();

        if (conflictingAppointment)
        {
            return BadRequest("Вибраний ветеринар зайнятий на цей час. Будь ласка,
            оберіть інший час або ветеринара.");
        }
        using (var transaction = await _context.Database.BeginTransactionAsync())
        {
            try
            {
                var appointment = new Appointment
                {
                    Client = client,
                    Pet = pet,
                    Veterinarian = veterinarian,
                    Service = service, // Service все ще прив'язується
                    AppointmentDateTime = dto.AppointmentDateTime
                };
                _context.Appointments.Add(appointment);
                await _context.SaveChangesAsync();

                int nextQueuePosition = 1;

                var lastQueueEntryForVet = await _context.Queues
                    .Where(q => q.Veterinarian.VeterinarianId ==
                        veterinarian.VeterinarianId)
                    .OrderByDescending(q => q.QueuePosition)
                    .FirstOrDefaultAsync();

                if (lastQueueEntryForVet != null)

```

```

        {
            nextQueuePosition = lastQueueEntryForVet.QueuePosition + 1;
        }

        var queueEntry = new Queue
        {
            Veterinarian = veterinarian,
            Appointment = appointment,
            QueuePosition = nextQueuePosition,

            Status = "Waiting"
        };

        _context.Queues.Add(queueEntry);

        await _context.SaveChangesAsync();
        await transaction.CommitAsync();

        return Ok(new
        {
            Message = "Запис на прийом та додавання до черги успішно
створено!",
            AppointmentId = appointment.AppointmentId,
            QueueId = queueEntry.QueueId,
            QueuePosition = queueEntry.QueuePosition
        });
    }

    catch (Exception ex)
    {
        await transaction.RollbackAsync();

        return StatusCode(500, $"Внутрішня помилка сервера: {ex.Message}");
    }
}

[HttpGet("available-slots")]

public async Task<IActionResult> GetAvailableTimeSlots(
[FromQuery] int veterinarianId,
[FromQuery] DateTime date)
{
    if (veterinarianId <= 0 || date == default)
    {
        return BadRequest("Необхідно вказати ID ветеринара та дату.");
    }

    var veterinarian = await _context.Veterinarians.FindAsync(veterinarianId);

    if (veterinarian == null)

```

```

    {
        return NotFound("Ветеринара не знайдено.");
    }

    if (date.Date < DateTime.UtcNow.Date)
    {
        return BadRequest("Не можна отримати доступні слоти для минулої дати.");
    }

    if (date.DayOfWeek == DayOfWeek.Saturday || date.DayOfWeek ==
DayOfWeek.Sunday)
    {
        return Ok(new List<AvailableSlotDto>());
    }

    var existingAppointments = await _context.Appointments
.Where(a => a.Veterinarian.VeterinarianId == veterinarianId &&
    a.AppointmentDateTime.Date == date.Date)
.OrderBy(a => a.AppointmentDateTime)
.Select(a => new
{
    Start = a.AppointmentDateTime,
    End = a.AppointmentDateTime.AddMinutes(DEFAULT_SERVICE_DURATION_MINUTES)
})
.ToListAsync();

    var availableSlots = new List<AvailableSlotDto>();
    var slotDuration = DEFAULT_SERVICE_DURATION_MINUTES;

    var currentSlotStart = date.Date.Add(WORK_START_TIME);

    while (currentSlotStart.AddMinutes(slotDuration) <=
date.Date.Add(WORK_END_TIME))
    {
        var currentSlotEnd = currentSlotStart.AddMinutes(slotDuration);
        bool isAvailable = true;

        foreach (var existingAppt in existingAppointments)
        {
            if (currentSlotStart < existingAppt.End && currentSlotEnd >
existingAppt.Start)

```

```

        {
            isAvailable = false;
            break;
        }
    }
    if (date.Date == DateTime.UtcNow.Date && currentSlotEnd <=
DateTime.UtcNow)
    {
        isAvailable = false;
    }
    if (isAvailable)
    {
        availableSlots.Add(new AvailableSlotDto
        {
            StartTime = currentSlotStart,
            EndTime = currentSlotEnd
        });
        currentSlotStart = currentSlotEnd;
    }
    return Ok(availableSlots);
}
}
}

```

AppointmentHistoryController

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Security.Claims;
using VetclinicSys.Server.Data;
using VetclinicSys.Shared.DTOS;

namespace VetclinicSys.Server.Controllers
{
    [Authorize(Roles = "Client")]
    [ApiController]
    [Route("api/[controller]")]
    public class AppointmentHistoryController : ControllerBase
    {
        private readonly AppDbContext _context;

        public AppointmentHistoryController(AppDbContext context)
        {
            _context = context;
        }

        [HttpGet]
    }
}

```

```

public async Task<IActionResult> GetHistory()
{
    var userIdString = User.FindFirstValue(ClaimTypes.NameIdentifier);

    if (string.IsNullOrEmpty(userIdString))
    {
        return Unauthorized("Не вдалося визначити ідентифікатор користувача.");
    }

    int userId;
    if (!int.TryParse(userIdString, out userId))
    {
        return Unauthorized("Некоректний ідентифікатор користувача.");
    }

    var history = await _context.Appointments
        .Where(a => a.Client.UserId == userId)
        .OrderByDescending(a => a.AppointmentDateTime)
        .Include(a => a.Pet)
        .Include(a => a.Veterinarian)
        .Include(a => a.Service)
        .Select(a => new AppointmentHistoryItemDto
        {
            AppointmentId = a.AppointmentId,
            AppointmentDateTime = a.AppointmentDateTime,
            PetName = a.Pet.Name,
            VeterinarianName = a.Veterinarian.Name,
            ServiceName = a.Service.Name
        })
        .ToListAsync();

    return Ok(history);
}
}

```

VeterinariansController

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using VetclinicSys.Server.Data;
using VetclinicSys.Shared.DTOs;
namespace VetclinicSys.Server.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class VeterinariansController : ControllerBase
    {
        private readonly AppDbContext _context;

        public VeterinariansController(AppDbContext context)
        {
            _context = context;
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<VeterinarianDto>>> GetVeterinarians()
        {
            var veterinarians = await _context.Veterinarians
                .Select(v => new VeterinarianDto
                {
                    Id = v.VeterinarianId,
                    Name = v.Name
                })
                .ToListAsync();

            return Ok(veterinarians);
        }
    }
}

```

```

        })
        .ToListAsync();

        return Ok(veterinarians);
    }
}

```

ServicesController

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using VetclinicSys.Server.Data;
using VetclinicSys.Shared.DTOS;

namespace VetclinicSys.Server.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class ServicesController : ControllerBase
    {
        private readonly ApplicationDbContext _context;

        public ServicesController(ApplicationDbContext context)
        {
            _context = context;
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<ServiceDto>>> GetServices()
        {
            var services = await _context.Services
                .Select(s => new ServiceDto
                {
                    Id = s.ServiceId,
                    Name = s.Name,
                    Price = s.Price,
                    Currency = s.Currency
                })
                .ToListAsync();

            return Ok(services);
        }
    }
}

```

VetclinicSys.Shared:

User.cs

```
namespace VetclinicSys.Shared.Models
{
    public class User
    {
        public int UserId { get; set; }

        public string Name { get; set; } = string.Empty;

        public string Email { get; set; } = string.Empty;

        public string PasswordHash { get; set; } = string.Empty;

        public string Role { get; set; } = string.Empty;

        public ICollection<Pet> Pets { get; set; }

        public ICollection<Appointment> Appointments { get; set; }
    }
}
```

Pet.cs

```
namespace VetclinicSys.Shared.Models
{
    public class Pet
    {
        public int PetId { get; set; }

        public string Name { get; set; } = string.Empty;

        public string Species { get; set; } = string.Empty;

        public int? AgeYears { get; set; }

        public User Owner { get; set; }

        public ICollection<Appointment> Appointments { get; set; }
    }
}
```

Appointment.cs

```
namespace VetclinicSys.Shared.Models
{
    public class Appointment
    {
        public int AppointmentId { get; set; }

        public User Client { get; set; }

        public Pet Pet { get; set; }

        public Veterinarian Veterinarian { get; set; }

        public Service Service { get; set; }

        public DateTime AppointmentDateTime { get; set; }
        public Queue Queue { get; set; }
    }
}
```

```
}  
Veterinarian.cs
```

```
namespace VetclinicSys.Shared.Models  
{  
    public class Veterinarian  
    {  
        public int VeterinarianId { get; set; }  
  
        public string Name { get; set; } = string.Empty;  
  
        public string Specialization { get; set; } = string.Empty;  
  
        public bool Available { get; set; } = true;  
  
        public ICollection<Appointment> Appointments { get; set; }  
  
        public ICollection<Queue> Queues { get; set; }  
    }  
}
```

Service.cs

```
namespace VetclinicSys.Shared.Models  
{  
    public class Service  
    {  
        public int ServiceId { get; set; }  
  
        public string Name { get; set; } = string.Empty;  
  
        public decimal Price { get; set; }  
  
        public string Currency { get; set; } = string.Empty;  
  
        public ICollection<Appointment> Appointments { get; set; }  
    }  
}
```

Queue.cs

```
namespace VetclinicSys.Shared.Models  
{  
    public class Queue  
    {  
        public int QueueId { get; set; }  
  
        public Veterinarian Veterinarian { get; set; }  
  
        public Appointment Appointment { get; set; }  
  
        public int QueuePosition { get; set; }  
  
        public string Status { get; set; } = string.Empty;  
    }  
}
```

StrikePlagiarism.com

© 2014-2025

ua

Україна

Дата звіту

6/14/2025

Дата редагування

⚠

Звіт не був оцінений

Звіт подібності

метадані

Назва організації

Kyiv National Economic University named after Vadym Hetman KNEU

Заголовок

РОЗРОБЛЕННЯ СИСТЕМИ ЕЛЕКТРОННОГО ЗАПИСУ ТА УПРАВЛІННЯ ЧЕРГОЮ ДЛЯ ВЕТЕРИНАРНОЇ КЛІНІКИ

Автор

Науковий керівник / Експерт

Кищук Аліна ВолодимирівнаМозгаллі О.П.

підрозділ

кафедра інформаційних систем в економіці

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

4.95%

4.95%

КП 1

0.95%

0.95%

КП 2

0.20%

0.20%

КЦ

25

Довжина фрази для коефіцієнта подібності 2

13952

Кількість слів

106068

Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		11
Інтервали		0
Мікропробіли		0
Білі знаки		1
Парафрази (SmartMarks)		63

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз

Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://fr.kneu.edu.ua/server/api/core/bitstreams/8886d85e-0996-456f-8b04-70b8d9883e0e/content	43 0.31 %
2	https://fr.kneu.edu.ua/server/api/core/bitstreams/8886d85e-0996-456f-8b04-70b8d9883e0e/content	31 0.22 %
3	https://fr.kneu.edu.ua/bitstreams/066dbca0-6afa-4cf4-a181-c3e88a2aab6f/download	31 0.22 %
4	https://vchika.in.ua/metodichni-rekomendaciyi-shodo-vikonannya-vipuskного-bakalavrs.html?page=2	28 0.20 %
5	https://fr.kneu.edu.ua/bitstreams/4559c31c-8da2-4bfa-b1bd-f336a89d8c4b/download	23 0.16 %