Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Системотехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка та дослідження інформаційної технології електронного документообігу ветеринарного закладу
(тема роботи)

Виконав: здобувач групи ІТПм-21-2
Баку А.М.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформаційні технології проектування
(повна назва освітньої програми)

Керівник доц. каф. СТ Петрова Р.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри системотехніки _____ Гребеннік І.В.
(підпис) (прізвище, ініціали)

2022 р.

Я як студент(ка) ХНУРЕ розумію і підтримую політику закладу із академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

17.12.2022 _____ Баку А.М.
(дата, підпис, прізвище студента/-ки)

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.

Керівник кваліфікаційної роботи _____ доц. Петрова Р.В.

Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.

Керівник кваліфікаційної роботи _____ доц. Петрова Р.В.

Попередній захист проведено 20 грудня 2022 р.

Керівник кваліфікаційної роботи _____ доц. Петрова Р.В.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Системотехніки

Рівень вищої освіти другий(магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформаційні технології проектування
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
« 20 » грудня 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Баку Андрію Миколайовичу
(прізвище, ім'я, по батькові)

1. **Тема роботи** Розробка та дослідження інформаційної технології електронного документообігу ветеринарного закладу
затверджена наказом університету від 21 листопада 2022 р. № 1504Ст
2. **Термін подання студентом роботи до екзаменаційної комісії** 21 грудня 2022 р.
3. Вихідні дані до роботи Дослідити та розробити компоненти інформаційної технології електронного документообігу на базі інформаційної системи роботи ветеианрного закладу. Перелік використовуваних програмних засобів: ОС Microsoft Windows 10, Visual Studio Code, MySQL Server/Workbench 8.0.25. Мова програмування додатку – Python 3.11.0, з використанням фреймворку Django та бібліотеки React. Технічне забезпечення: IBM-сумісний пристрій з ЦП Intel Core i3/AMD аналог та вище, 8 Гб оперативної пам'яті та об'ємом жорсткого диску не менше 256Гб.
4. **Перелік питань, що потрібно опрацювати в роботі** 4.1 Вступ 4.2 Аналіз предметної області. 4.2.1 Дослідження інформаційних веб-систем. 4.2.2 Аналіз предметної області, що визначає діяльність компанії. 4.2.3 Аналіз існуючих аналогів розроблюваної системи. 4.2.4 Визначення сфери застосування розроблюваної системи. 4.2.5 Аналіз виявлених проблем діяльность ветеринарних закладів. 4.2.6 Постановка задачі дослідження. 4.3 Огляд методів та технологій, які застосовуються в предметній області. 4.3.1 Технології електронного документообігу. 4.3.2 Технології реалізації клієнтської частини додатку. 4.3.2.1 Огляд технології HTML. 4.3.2.2 Огляд технології CSS. 4.3.2.3 Огляд технології JavaScript. 4.3.2.4 Огляд технології React. 4.3.3 Технології реалізації серверної частини застосунку 4.3.3.1 Огляд технології Python 4.3.3.2 Огляд технології Django 4.3.4 Вибір середовища розробки веб-застосунку клініки 4.3.5 Огляд технологій взаємодії клєнта та серверу 4.3.6 Вибір системи управління базами даних проектованого застосунку 4.3.7 Висновки до технологій що застосовуються при розробці 4.4 Розробка вимог до створюваної ІС. 4.4.1 Розробка системних вимог до створюваної інформаційної технології електронного документообігу ветеринарного закладу. 4.4.2 Опис архітектури розроблюваної системи. 4.4.3 Розробка функціональним вимог до інформаційної технології електронного документообігу ветеринарного закладу. 4.4.4 Розробка діаграми варіантів використання інформаційної технології електронного документообігу ветеринарного закладу. 4.4.5 Розробка моделі потоків даних інформаційної технології електронного документообігу ветеринарного закладу. 4.5 Опис прийнятих проектних рішень при розробці інформаційної технології. 4.5.1 Логічне й фізичне

моделювання даних розроблюваної технології. 4.5.2 Генерація бази даних розроблюваної системи. 4.5.3 Розробка алгоритму роботи інформаційної технології на базі веб-додатку. 4.5.4 Генерація веб-додатку 4.5.5 Генерація інформаційної технології електронного документообігу.

5. **Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій** 5.1 Контекстна діаграма (1 аркуш формату А4). 5.2 Декомпозиція контекстної діаграми (1 аркуш формату А4). 5.3 Діаграма варіантів використання (1 аркуш формату А4). 5.4 Контекстна діаграма моделі потоків даних (1 аркуш формату А4). 5.5 Взаємодія елементів ІС (1 аркуш формату А4). 5.6 Схема електронного документообігу у ветеринарному закладі (1 аркуш формату А4). 5.7 Логічна структура даних інформаційної системи (1 аркуш формату А4). 5.8 EER-модель бази даних ІС (1 аркуш формату А4). 5.9 Базовий алгоритм роботи системи (1 аркуш формату А4). 5.10 Базова схема доступу до сторінок веб-додатку (1 аркуш формату А4). 5.11 Базове представлення клієнтської електронної карти. 5.12 Первинний вигляд сторінок для вводу даних. 5.13 Загальний вигляд направлення на прийом (1 аркуш формату А4). 5.14 Аналіз конкурентів системи (1 аркуш формату А4).

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Отримання завдання атестаційної роботи	21.11.22	Виконано
2.	Проведення аналізу предметної галузі системи	21.11.22 – 23.11.22	Виконано
3.	Аналіз виявлених проблем обраної системи	24.11.22	Виконано
4.	Постановка задачі на розробку системи	26.11.22	Виконано
5.	Вибір технологій для реалізації веб-додатку	27.11.22 – 28.11.22	Виконано
6.	Створення бази даних додатку	29.11.22 – 30.11.22	Виконано
7.	Розробка інформаційної технології вирішення задачі	01.12.22 – 03.12.22	Виконано
8.	Опис прийнятих проектних рішень при розробці технології	04.12.22 – 05.12.22	Виконано
9.	Генерація бази даних	06.12.22	Виконано
10.	Розробка алгоритму роботи інформаційної технології	07.12.22	Виконано
11.	Генерація веб-додатку та інформаційної технології	08.12.22 – 11.12.22	Виконано
12.	Оформлення пояснювальної записки	12.12.22 – 15.12.22	Виконано
13.	Опис графічних матеріалів та програмної документації	16.12.22 – 19.12.22	Виконано
14.	Представлення на рецензування	18.12.22	Виконано
15.	Представлення роботи до ДЕК	21.12.22	Виконано

Дата видачі завдання 21 листопада 2022 р.

Студент _____
(підпис)

Керівник роботи _____ к.т.н. Петрова Р.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка магістерської кваліфікаційної роботи: 82 с., 0 табл., 16 рис., 3 додатки , 42 джерел інформації.

ВЕБ-ДОДАТОК, MYSQL, JAVASCRIPT, PYTHON, ІНФОРМАЦІЙНА СИСТЕМА, ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, DJANGO, IDEF0

Об’єктом дослідження даної роботи є процес автоматизації документообігу ветеринарного закладу за рахунок створення електронного документообігу й впровадження його в процес реєстрації, авторизації, оформлення запису на прийом, обробки запису та ведення звітності ветеринарного закладу.

Предметом досліджень є програмні методи створення інформаційної технології на базі інформаційної системи з клієнтською і серверною частиною веб-додатку, що дозволить автоматизувати процес обліку записів на прийом до ветеринарного закладу.

Мета роботи – розробка та дослідження інформаційної технології електронного документообігу ветеринарного закладу.

Методи дослідження – функціональний підхід, методи структурного аналізу і моделювання реляційних баз даних, методи проектування клієнт-серверних додатків для Інтернет-мереж, методи створення функціональних додатків.

У роботі проведено аналіз предметної області, яка належить до основних аспектів роботи ветеринарного закладу. Для вибору методів розробки проводиться аналіз існуючих варіантів і вибір кращого з існуючих для реалізації системи. Для визначення основних кроків та задач автоматизації проводиться постановка задачі, яка в повній мірі описує наявні проблеми в сучасному підході до функціонування ветеринарних закладів та варіанти виправлення даних проблем. Для визначення та уточнення вимог до створюваної системи проводиться функціональне моделювання системи та розробляються основні види UML-моделі, серед яких можна виділити «юз-кейс» діаграму та діаграму моделі потоків даних.

Область застосування системи – введення створюваної інформаційної технології електронного документообігу в ветеринарний заклад, який має старий спосіб роботи з документообігом, що представлений у паперовому вигляді. Таке впровадження дозволить автоматизувати роботу закладу, націлившись на покращення аспектів роботи з клієнтами.

ABSTRACT

Explanatory note of the master's qualification thesis: 82 pages, 0 tables, 16 figures, 3 appendices, 42 sources of information.

WEB APPLICATION, MYSQL, JAVASCRIPT, PYTHON, INFORMATION SYSTEM, INFORMATION TECHNOLOGY, DJANGO, IDEF0

The object of research of this work is the process of automating the document flow of a veterinary institution due to the creation of electronic document flow and its implementation in the process of registration, authorization, registration of an appointment, processing of records and reporting of a veterinary institution.

The subject of research is software methods of creating information technology based on an information system with a client and server part of a web application, which will allow automating the process of accounting for appointments to a veterinary institution.

The purpose of the work is the development and research of the information technology of the electronic document flow of the veterinary institution.

Research methods - functional approach, methods of structural analysis and modeling of relational databases, methods of designing client-server applications for Internet networks, methods of creating functional applications.

The work analyzes the subject area, which belongs to the main aspects of the work of a veterinary institution. For the selection of development methods, an analysis of existing options is carried out and the best of the existing ones is selected for the implementation of the system. To determine the main steps and tasks of automation, a problem statement is made, which fully describes the existing problems in the modern approach to the functioning of veterinary institutions and options for correcting these problems. To define and clarify the requirements for the system being created, functional modeling of the system is carried out and the main types of UML models are developed, among which the "use-case" diagram and the diagram of the data flow model can be distinguished.

The field of application of the system is the introduction of the newly created information technology of electronic document management in a veterinary institution that has an old way of working with document management, which is presented in

paper form. This implementation will allow automating the work of the institution, aiming at improving aspects of working with clients.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування.

UML – уніфікована мова моделювання (Unified Modeling Language) - це система позначень, яку можна застосовувати для об'єктно-орієнтованого аналізу і проектування.

JS – повноцінна динамічна мова програмування, яка, у застосуванні до HTML документу, може надати динамічну інтерактивність на веб-сайтах.

MySQL – система управління реляційними базами даних.

CSS – каскадні таблиці стилів.

URL – уніфікований покажчик інформаційного ресурсу –адресу, яка видана унікальному ресурсу в інтернеті.

DOM – об'єктна модель документа – надає структуроване подання документа та визначає те, як ця структура може бути доступна з програм, які можуть змінювати вміст, стиль та структуру документа.

ВСТУП

Тематикою даної роботи є розробка та дослідження інформаційної технології електронного документообігу ветеринарного закладу на базі інформаційної системи, тобто мається на увазі інтерактивний веб-додаток, який міститиме функціонал електронного документообігу, який на базі користувацьких даних дозволить створювати електронні документи та відмовитись від ведення паперового документообігу. Інформаційну систему можна сприймати як інтерактивний веб-додаток для ведення електронної комерції послуг, що надаються ветеринарним закладом.

Актуальність проекту полягає в тому, що більшість наявних ветеринарних закладів звикли до роботи з паперовими документами. Однак, використання інформаційної технології електронного документообігу дозволить значно прискорити процес роботи закладів такого типу, за рахунок повної відмови від паперових документів та повним переходом на електронні документи, що в свою чергу знімає навантаження на адміністраторів ветеринарного закладу. Ті, в свою чергу, отримають зручний функціонал роботи зі звітністю ветеринарного закладу. Не менш актуальною перевагою створюваної системи є можливість віддаленого запису на прийом, що дозволить залучити до даної процедури більшу кількість людей, які звикли вирішувати свої проблеми суто в телефонному форматі.

Мета роботи – підвищення ефективності роботи ветеринарних закладів за рахунок створення інформаційної технології електронного документообігу, що дозволить користувачам і працівникам ветеринарного закладу насолоджуватися процесом запису на прийом, обробки оформлених записів на прийом та власне процесом виконання прийому з подальшим введенням необхідних даних.

Завдання дослідження – дослідити та виявити основні концепції до підходу електронного документообігу у сфері ветеринарної діяльності. Сформувати основні проблеми застарілого підходу до документообігу.

Предметом дослідження виступають програмні методи створення інформаційних систем та технологій з серверною і клієнтською частиною, що дозволять автоматизувати процес ведення обліку записів на прийом до ветеринарного закладу.

Об’єктом дослідження кваліфікаційної роботи виступає процес автоматизації технології паперового документообігу ветеринарного закладу на сучасний, що представлений у вигляді електронного документообігу.

Методи дослідження – методи структурного аналізу системи і моделювання реляційних баз даних, функціональний підхід до створення інформаційних систем, методи проектування клієнт-серверних веб-додатків, методи генерації шаблонів для ведення документації.

Одержані результати слід вважати науково новими, адже значна кількість сучасних ветеринарних закладів звикла покладатись на паперовий документообіг і навіть не замислюється про ідею електронного документообігу. Ознайомлення з проблематикою та методами і підходами до вирішення описаних проблем дозволить ветеринарним закладам вийти на новий рівень надання своїх послуг. При більш комплексному програмному підході до вирішення поставленої задачі буде отримано неймовірну інформаційну технологію, що дозволить користувачам ветеринарних закладів з нової точки зору подивитись на ветеринарні заклади.

При підготовці кваліфікаційної роботи була підготовлена та опублікована наукова робота: «Дослідження інформаційної технології електронного документообігу для впровадження до роботи системи ветеринарного закладу» у рамках IX Міжнародної науково-практичної конференції «MODERN RESEARCH IN WORLD SCIENCE» у м. Львів, 28-30 листопада, 2022 [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження інформаційних веб-систем

Автоматизація процесів у ветеринарній діяльності має велике значення. Адже вона дозволить керівництву компанії зосередитися на більш важливих аспектах роботи, таких як якість послуг, що надаються клієнтам, та забезпечення задоволення потреб спеціалістів клініки. Власне, створювана інформаційна веб-система і покликана допомогти з цим.

Інформаційна система, у загальному представляє собою програмну систему, яка націлена на виконання головних інформаційних процесів виробництва, з-поміж яких можна виділити головні – пошук, збирання, опрацювання та зберігання необхідної інформації.

Інформаційні системи можна класифікувати наступним чином:

- ручні; в системах даного типу опрацювання інформації у загальному форматі покладено на людину;
- автоматизовані; системи, які передбачають поділ обов’язків по роботі з інформацією між застосунком та людиною;
- автоматичні; системи, які обходяться без участі людини, тобто усі функції над інформацією здійснюються автоматично, за допомогою технічних засобів.

Головними завданнями інформаційних систем слід вважати наступні:

- виявлення, збір, збереження, обробка та функціонал видачі інформації, яка характеризує стан виробництва;
- розподіл інформації між головними частинами виробництвами, тобто керівниками, підрозділами та виконавцями різних типів робіт, в залежності від їх ролі у системі.

Сучасні інформаційні системи для своєї роботи використовують функціонал електронного документообігу, під яким слід розуміти загальну сукупність інформації, яка створюється та циркулює на підприємстві. Дана інформація може поступати ззовні або створюється спеціально для надання за межі виробництва.

Електронний документообіг, як технологія включає зовнішні та внутрішні документи виробництва [2]. Відмінність такого підходу від звичайного

документообігу лише в тому, що передбачено створення електронних документів – документів, інформація у яких фіксується шляхом запису до них електронних даних.

Головною ознакою електронного документообігу слід вважати факт створення електронних документів замість використання паперових з метою накопичення, надання, обробки інформації, а також процесу її обміну [3]. Під електронним документообігом розуміють наступні етапи роботи з електронними документами:

- створення документів;
- обробка документів;
- передача документів між різними елементами виробництва;
- використання документів;
- зберігання документів;
- знищення документів.

Веб-застосунок можна описати, як програмне забезпечення, доступ до якого можна отримати з будь-якого браузеру. Тобто, застосунок, який доступен майже з будь-якого електронного девайсу. Веб-застосунок складається з клієнтської та серверної частини [4]. Клієнтська частина, частіше за все, написана на мовах для веб-програмування, таких як HTML, CSS та JavaScript [5]. Серверна ж частина для своєї основи може використовувати будь-яку з мов програмування або фреймворків, типу Python, Java, тощо.

Розрізняють наступні види веб-застосунків [6]:

- PWA (прогресивний веб-застосунок); може нагадувати собою мобільний застосунок, тобто досягається неймовірна гнучкість роботи на пару зі зручним інтерфейсом користувача; у деяких випадках дані застосунки дозволяють часткову роботу без інтернету, у випадку роботи з інформаційною стороною застосунку;
- HTML-5 (звичайна веб-сторінка); застосунок, який імітує собою програму; у більшості підходить для створення невеликих систем, від яких не очікується виконання важких функцій, по типу доступу до бази даних;
- SPA (одно сторінковий застосунок); представляє собою застосунок із динамічним оновленням веб-сторінок; працює таким чином, що статична основа веб-сайту залишається незмінною, однак дані введені

користувачем оновлюються при взаємодії, тобто без необхідності ручного оновлення сторінок (F5); технологія використовується для створення адаптивних застосунків.

Існує класифікація веб-застосунків на основі їх потенційного призначення:

- E-commerce системи; системи, що створюються для відтворення онлайн замовлень та продажів товарів без сторонніх осіб; яскравими представниками цього типу виступають інтернет-магазини;
- CRM-системи; даний тип систем призначений автоматизувати відділ продаж компанії та всіх робочих заявок, які надходять до системи;
- ERP-системи; системи, що покликані автоматизувати не лише відділ продаж компанії, а усі підрозділи компанії, тобто передбачає повну автоматизацію роботи системи;
- корпоративні портали; даний тип систем можна описати, як інформаційні застосунки для працівників фірми, за допомогою якого проводяться інформування співробітників, їх контроль, тощо.

Процес роботи веб-застосунка не є важким, його можна описати у декілька кроків. По-перше, користувач своїми діями створює запит, який відправляється на веб-сервер, відправка відбувається за допомогою URL-запиту, чи можливостями веб-застосунку. Даний запит надсилається до серверу, де проходять необхідні перетворення даних та збереження отриманих даних в БД. Фінальним кроком є формування запиту-відповіді на дії користувача, які надсилаються до клієнтської частини системи, для виведення на екран користувача. Даний процес є циклічним, допоки користувач користується веб-застосунком.

Веб-застосунок можна назвати дуже гарним способом для досягнення цілей, адже він містить в собі багато переваг, порівняно зі стандартними системами, з яких можна виділити головні – веб-застосунки підтримуються на будь-якій ОС, яка може надати доступ до браузеру; легша система підтримки фінальної версії застосунку; легкість програмування, так як немає необхідності взаємодії з основними елементами комп’ютера, такими як процесор та його ядра [7].

Беручи до уваги написане вище маємо наступне – необхідно розробити та дослідити роботу веб-застосунку ветеринарного закладу. Тобто створити веб-застосунок, який дозволить автоматизувати основні процеси роботи ветеринарного закладу, до яких відносяться наступні – ведення обліку пацієнтів

клініки, оформлення та обробка записів на прийом до спеціалістів клініки, ведення звітності з роботи спеціалістів. Досліджувану систему можна віднести до SPA-застосунку.

1.2 Аналіз предметної області, що визначає діяльність компанії

Предметна область завдання – інформаційна система ветеринарного закладу, яка представлена специфічною формою інтерактивного веб-застосунку. Веб-застосунок призначений для автоматизації роботи клініки, тобто повного її відходження від паперової роботи. Таким чином застосунок бере на себе всю роботу не-робочого персоналу клініки, починаючи від створення особистих справ пацієнтів і завершуючи створенням записів на прийом до закладу [8].

Ветеринарна справа виступає однією із найрозповсюдженіших наукових знань у сучасному світі, адже кількість домашніх улюбленців з кожним днем зростає у арифметичній прогресії, паралельно з кількістю хвороб та способів їх лікування.

Основними функціями та завданнями ветеринарної справи у світі прийнято вважати наступні:

- оновлення та створення нових зоологічних і ветеринарно-санітарних умов для утримання домашніх тварин;
- створення та утвердження обов’язкових норм матеріального забезпечення ветеринарних заходів;
- узгодження системи і порядку фінансування ветеринарної діяльності;
- визначення порядку і критеріїв оцінки результатів діагностичних досліджень та норм витрат праці ветеринарних спеціалістів на виконання ветеринарних робіт.

Ветеринарним закладом прийнято вважати лікувально-профілактичний заклад для надання ветеринарної допомоги хворим тваринам на прийомі у спеціалізованого лікаря.

Метою створення та функціонування таких закладів є надання індивідуально-лікувальних послуг, спрямованих на лікування та догляд за домашніми тваринами, для отримання з цього прибутку.

Законодавство визначає такі нормативні документи, які регламентують діяльність ветеринарних закладів – Конституція України, а саме Закон «Про ветеринарну медицину», постанови Верховної Ради та Кабінету Міністрів та інші урядові акти, що регламентують організації ветеринарної медицини, а також видані органами управління ветеринарною службою положення , інструкції, правила, встановлення та вказівки. Основу Закону складатимуть загальні правила охорони тварин від хвороб, а також найважливіші положення організації ветеринарних заходів.

Організаційні можливості діяльності закладу у сфері електронної комерції визначає Закон України «Про електронну комерцію». Цей документ визначає організаційно-правові засади здійснення в Україні електронної комерції, встановлює чіткі правові вимоги та механізми дистанційного укладення та виконання угод з застосуванням електронних інформаційно-комунікаційних засобів та технологій.

Кожен з працівників ветеринарного закладу зобов'язаний підпорядковуватися Закону України «Про ветеринарну медицину» та обов'язковим загальнодержавним ветеринарно-санітарним правилам і нормам.

Штат клініки складається з головного ветеринарного лікаря, директора, який у критичних випадках може виступати ветеринарним лікарем та виконувати обов'язки головного лікаря під час лікарняного чи відпустки останнього, чотирьох добових лікарів зміни, двох асистентів, одного старшого адміністратора та двох добових адміністраторів [9]. За необхідності, ветеринарний заклад може містити стажерів.

Ветеринарні лікарі, перелічені вище, мають вищу освіту за кваліфікацією «Ветеринарний лікар», та стаж роботи за кваліфікацією не менше 3 років. Обов'язковим є досвід роботи з дрібними домашніми тваринами та додаткові документи, що вказують на пройдені курси з підвищення кваліфікації.

Типова схема організаційної структури закладу наведена на рис.1.1. Дана схема організаційної роботи [10] дозволяє опиратись на майбутній комфортний потік роботи закладу, розраховуючи на більше ніж 30 клієнтів за добу [11].

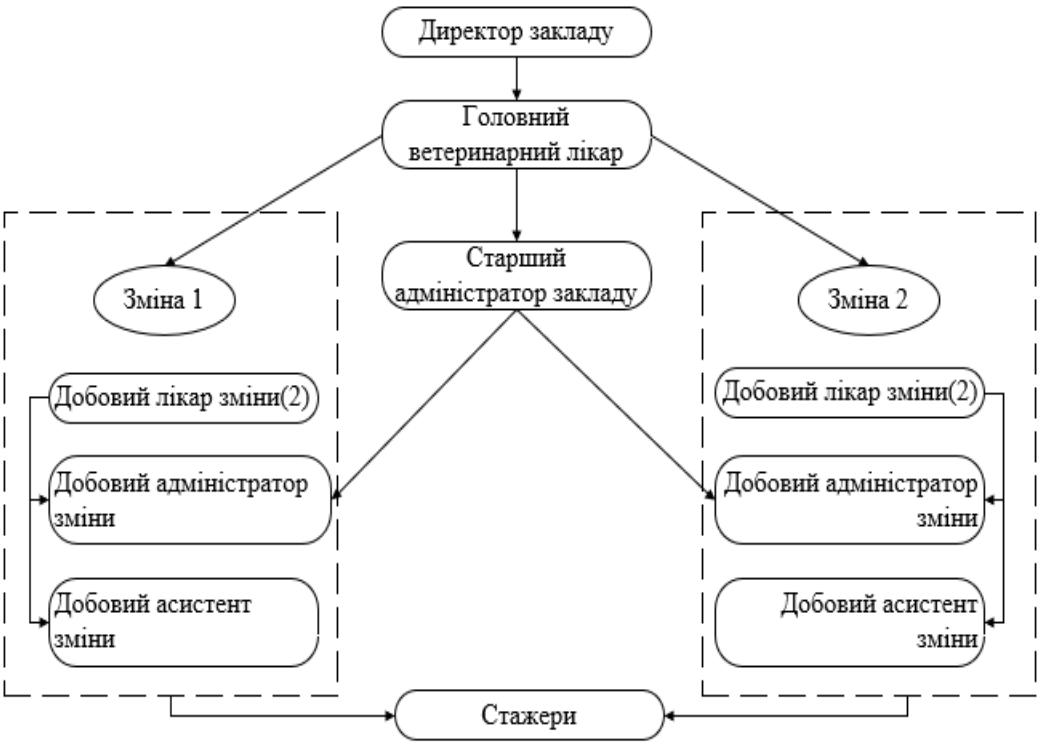

Рисунок 1.1 – Типова схема організаційної структури закладу

Кожен із наведених типів робітників ветеринарної клініки має свій функціонал у системі.

Директор закладу «малює» своє бачення веб-застосунку, надає ідеї по реалізації БД, так як дуже добре розуміється у даній предметній області. У час, коли він замінює головного ветеринарного лікаря він виступає безпосередньо користувачем системи. Адже він приймає клієнтів ветеринарної клініки і після проведення прийому він зобов’язаний заповнити дані прийому, тобто ввести інформацію про проведену роботу, та завершити запис на прийом. Директор закладу також має доступ до адміністративної панелі застосунку, функціонал якої буде описаний далі.

Головний ветеринарний лікар, має безпосередній доступ до системи, як управляючий виробництвом та спеціаліст, який надає послуги клієнтам ветеринарного закладу. Він має наступні функції у системі:

- авторизація у системі за раніше створеними обліковими записами адміністратора;
- доступ до даних прийомів;
- доступ до даних користувачів системи;
- внесення і оновлення даних прийому;

- перегляд звітної інформації ветеринарного закладу;
- доступ до адміністративної сторінки системи;
- вихід з облікового запису адміністратора.

Авторизація у системі є процесом входу спеціаліста до системи, за допомогою даних облікового запису адміністратора, що надасть йому можливості роботи з системними даними прийомів.

Доступ до даних прийомів можна описати як можливість спеціаліста відслідковувати необхідну інформацію по прийомам, таку як час прийому, та клієнтів.

Доступ до даних користувачів системи бажано розуміти як можливість головного лікаря мати постійний доступ до електронних карток клієнтів фірми, та їх тварин.

Внесення і оновлення даних прийому необхідно розуміти як процес внесення даних прийому до бази даних. Дані включають інформацію по прийому, таку як отриманий висновок роботи, дата наступного огляду та завершення прийому.

Перегляд звітної інформації закладу є функцією системи, яка надає доступ користувачу до системних звітів, серед яких можна виділити добові та тижневі звіти адміністраторів закладу з його роботи. Також, звіти можуть містити інформацію щодо потреб закладу у лікарському та матеріальному забезпеченні.

Доступ до адміністративної сторінки системи є функцію, яка надає користувачеві доступ до «адмін-панелі» застосунку. Тобто, користувач з такою можливістю має доступ до додавання певних даних до інформаційної системи.

Вихід з облікового запису є функцією виходу з системи, що дозволяю відслідковувати час роботи спеціалістів.

Добові лікарі зміни виконують спрощені дії, порівняно з головним лікарем закладу. Адже головною їх метою є провести прийом, та ввести післяприйомні дані до електронної картки тварини. Добовий лікар має наступні функції у інформаційній системі:

- авторизація у додатку;
- доступ до даних користувачів клініки;
- доступ до даних прийомів;
- можливість внесення та оновлення даних запису;
- можливість формування наступного запису на прийом;

- вихід з облікового запису.

Можливість внесення та оновлення даних запису є можливістю спеціаліста взаємодіяти із додатком, тобто вносити інформацію, яку було визначено на прийомі, курс лікування, тощо.

Можливість формування наступного запису на прийом – можливість ветеринарного лікаря за допомогою адміністративного інтерфейсу створювати у системі запис на прийом на довільну дату. Тобто, можливість призначення наступного прийому на бажану дату.

Старший адміністратор закладу має повноцінний доступ до бази даних, адже він відповідає за процес налагодження трудового кругообігу у системі. Його головними функціями у системі є наступні:

- авторизація у системі;
- робота з електронною поштою закладу;
- створення тижневих звітів роботи закладу;
- впровадження розкладу роботи спеціалістів;
- налагодження роботи з зовнішніми поставниками;
- створення електронних документів закладу;
- вихід з облікового запису.

Робота з електронною поштою з боку головного адміністратора є процес взаємодії з поштою для процесу контролю виконання обробки електронних записів системи.

Створення тижневих звітів роботи закладу можна описати як аналіз денних звітів добових адміністраторів та створення на основі їх результатів тижневого звіту з роботи закладу. Дані звіти надсилаються до головного лікаря закладу.

Впровадження розкладу роботи спеціалістів є функцією взаємодії з адміністративною панеллю для редагування інформації щодо розкладу роботи. Також передбачає власне процес аналізу результатів роботи спеціалістів із денних звітів для створення більш вдалого розкладу.

Налагодження роботи з зовнішніми поставниками є підфункцією роботи з електронною поштою, тобто за допомогою пошти адміністратор закладу знаходить нові канали для роботи з поставниками ліків та медичного устаткування.

Створення електронних документів є функцією, яка є головною для впровадження електронного документообігу у клініці. Під нею розуміється

виконання комплексу дій, які є базою для електронного документообігу. Тобто, створення, редагування, знищення, тощо.

Добовий адміністратор зміни, на відміну від старшого адміністратора закладу не виконує складну роботу з процесами трудового кругообігу у системі. Головною його функцією є обробка та контроль узгодження оформлених записів на прийом. До його системних функцій належать наступні:

- авторизація у системі;
- робота з електронною поштою;
- обробка оформлених записів на прийом;
- створення добових звітів роботи закладу;
- робота з реєстрацією клієнтів закладу;
- вихід з облікового запису.

Робота з електронною поштою, на відміну від головного адміністратора, передбачає роботу з створеними заявками на прийом. Робота з заявками виникає у випадку вибору користувачами варіанту для наступного зв’язку з адміністратором для уточнення аспектів запису на прийом.

Обробка оформлених записів на прийом необхідна для ведення процесу підтвердження оформлених записів на прийом. Тобто, присутня функція зв’язку з клієнтами, за необхідності, для уточнення даних запису.

Створення добових звітів роботи закладу є функцією, яка надає доступ до інтерфейсу створення звітів роботи закладу за день. Звіти надходять до старшого адміністратора для аналізу й обробки.

Робота з реєстрацією клієнтів закладу є функцією, яка за необхідності надає добовому адміністратору доступ до панелі реєстрації користувача та його тварини для наступного оформлення запису на прийом. Така необхідність вникає у разі неможливості користувачем оформлення онлайн запису на прийом.

Стажери в даному випадку не мають прямого доступу до системи, їх прямим обов’язком є присутність на прийомах та можлива допомога спеціалісту під час його роботи. У випадку необхідності вони можуть заповнювати інформацію, однак заповнення інформації буде вестись з облікового запису лікаря та під його наглядом.

1.3 Аналіз існуючих аналогів розроблюваної системи

Нині доступні аналоги розроблюваної системи представляють собою звичайну інформаційну веб-сторінку клініки. Вони знайомлять клієнта з основної інформацією про клініку – послугами, цінами, фахівцями, які здійснюють прийоми, різноманітні блоги, новини, тощо. Всі аналоги містять доволі приємний інтерфейс сторінок, багато високоякісних зображень тварин, які привертають до них увагу.

Однак, всі аналоги системи містять лише один функціонал, який представляє собою «доісторичну» форму для зв'язку з клінікою. Дана форма, у своїй більшості, містить три головних поля – Ім'я, Телефон та Опишіть вашу проблему [12]. Вона представляє собою своєрідне листування з адміністратором клініки. Йому на пошту приходить письмо з інформацією, причому іноді поля не містять валідації даних, тому інформацію, що до нього надходить може не відповідати дійсності. Після отримання письма адміністратор повинен звірити дані у письмі, перечитати дані, наведені у полі «Опишіть вашу проблему», на свій розсуд вибрати лікаря, та після цього виконувати зв'язок з клієнтом.

Таке рішення дуже позитивно впливає на бюджет сайту. Однак даний функціонал відбиває будь-яку автоматизацію роботи, адже адміністратор повинен провести величезну кількість роботи перед зв'язком з клієнтом. Клієнти, при такому підході не мають жодної структуризації, тобто веб-сайт не запам'ятовує даних клієнта, що знову ж так створює проблему кожен раз набирати одні й ті ж особисті дані, що негативно впливає на задоволення клієнта.

Клієнти закладу обмежені в виборі, що є найбільшою проблемою. Адже багатьом клієнтам хочеться один раз ввести дані запису, та більше не повертатися до цього.

Однак, переважна меншість сайтів має функціонал дещо наближений до створюваної системи. Він включає повноцінний етап реєстрації з послідуючою авторизацією. Що допомагає клієнтові економити свій час при вводі користувальницьких даних. Також, даній меншості притаманні більш продумані форми для запису на прийом, які іноді включають у себе поля вибору часу, та послуги. Однак і дану перевагу можна назвати недоліком, адже вибір часу зображено у вигляді абстрактних проміжків, типу «Вранці», «Після обіду», тощо. Є й застосунки, які дозволяють обрати тип послуги, яку необхідно виконати на

прийомі. Однак є одне велике «АЛЕ», немає сайту(застосунку) який містив би у собі сукупність перелічених вище функцій, тобто веб-сайти приймають розрізнений функціонал, один має час, другий реєстрацію і так далі.

Тому, з упевненістю можна сказати, що застосунок прийнятий на розробку буде унікальним у своєму роді, адже міститиме всі функціональні можливості, які повинен містити бездоганний застосунок, який орієнтований на задоволення клієнтів, та адміністратору закладу [13].

1.4 Визначення сфери застосування розроблюваної системи

Зважаючи на класифікацію інформаційних систем, обрана система є системою електронної комерції типу B2C, яка може бути описана забезпеченням електронних комерційних відносин між закладом-надавачем послуг та споживачами, які представлені клієнтами-власниками тварин.

Орієнтуючись на опис предметної області можна виділити основний список бізнес-процесів, притаманних системі:

- реєстрація у застосунку;
- авторизація у застосунку;
- оформлення запису на прийом;
- оновлення/додавання даних до кінцевої системи;
- зміна статусу запису на прийом;
- створення звітів роботи закладу;
- процедура оповіщення клієнта про підтвердження/відміну запису на прийом.

Реєстрація на сайті – бізнес-процес, що полягає у вводі клієнтом системи початкових даних, які у майбутньому будуть збережені у базі даних системи. Після збереження клієнт зможе проводити повноцінну авторизацію у застосунку. Тобто, після реєстрації у клієнта відпадає необхідність постійно вводити особисті дані, типу ім'я, телефон, електронна пошта, тощо.

Авторизація у застосунку – бізнес-процес, що орієнтований на можливості клієнтом авторизуватися у системі. Тобто, отримувати доступ до клієнтського кабінету, та функції запису на прийом до закладу.

Оформлення запису на прийом – процес, що полягає у вводі клієнтом даних тварини, виборі потенційного типу послуг, які необхідно вирішити на

прийомі, введенні бажаної дати, та виборі задовольняючого часу прийому. Присутня можливість вибору пункту, який відміняє необхідний зв'язок з адміністратором клініки. Після введення даних, вони зберігаються у БД, та надсилаються на пошту адміністратора, який при необхідності проводить уточнюючий зв'язок з клієнтом.

Оновлення/додавання даних до кінцевої системи – процес, який полягає у можливостях інтерфейсу застосунку масштабуватися у відповідності до даних закладу, які вносяться до бази даних.

Зміна статусу запису на прийом – функціонал серверу застосунку, який у відповідності до введених даних ставить відмітку статусу на прийом: «Підтверджений», «Скасований», «На розгляді».

Створення звітів роботи закладу – бізнес-процес, який передбачає можливість створення звітів про роботу закладу, з даних наявних у базі даних. Дана функція доступна адміністратору застосунку, представлена фунціоналом вибору двох дат, між якими необхідно створити звіт.

Процедура оповіщення клієнта про підтвердження/відміну запису на прийом – бізнес-процес, що виконує унікальний функціонал зі створення повноцінного веб-листа з повною інформацією про запис на прийом, який надходить користувачеві на пошту після підтвердження/скасування запису на прийом.

Маючи опис предметної області та основні бізнес-процеси, які притаманні даній області можна визначити список основних бізнес-функції системи:

- вхід до облікового запису користувача;
- вихід з облікового запису користувача;
- відображення загально-інформаційних даних закладу у інтерфейсі застосунку;
- перегляд списку записів закладу;
- сортування списку записів;
- додавання даних до системи;
- валідація полів форми запису на прийом;
- створення письма-підтвердження запису;
- надсилання письма-підтвердження на пошту користувачеві;
- формування звіту діяльності клініки.

Вхід до облікового запису користувача – функція, яка передбачає собою авторизацію клієнта, тобто розширення функціоналу його дій. При вході клієнта відбувається взаємодія у серверній частині застосунку, та користувачеві надаються індивідуальні дані для авторизації, для цього використовуватимуться cookie.

Вихід з облікового запису – виступає повною протилежністю попередній функції, адже передбачає вихід з системи, тобто усунення можливості роботи з даними користувача та формою запису на прийом.

Відображення загально-інформаційних даних закладу у інтерфейсі застосунку – функція, яка виконує вибірку інформаційних даних з бази даних, для наступного їх «вмонтування» у інтерфейс системи, тобто створення живого інтерфейсу з використанням випадаючих списків, та живого навігаційного меню системи.

Перегляд списку записів закладу – бізнес-функція, яка надає вибірку даних, представлених записами на прийом. Дані беруться з уведених раніше дат, тобто початкової та кінцевої дати.

Сортування списку записів – функція, що полягає у можливості сортувати записи, які були надані вибіркою. Сортування може відбуватися за полями наявними у вибірці.

Додавання даних до системи – функція, що доступна адміністратору закладу. Передбачає можливість додавання інформаційних даних закладу до бази даних, за наступним автоматичним оновленням інтерфейсу системи.

Валідація полів форми запису на прийом – функція, яка виконує стандартну валідацію обраних полів у формі оформлення запису на прийом. Присутня нестандартна валідація, тобто валідація для складних полів, по типу номеру телефона чи електронної пошти користувача.

Створення письма-підтвердження запису – бізнес-функція серверної частини застосунку, яка направлена на створення веб-письма підтвердження/скасування запису. Письмо створюється у форматі .html, з наступним збереженням у .pdf.

Надсилання письма-підтвердження на пошту користувачеві – функція вибірки даних електронної пошти клієнта, з наступним надсиланням створеного письма. До письма на пошті автоматично додається його тема, з вказанням прізвища клієнта, для більш вдалої орієнтації на пошті.

Формування звіту діяльності клініки – бізнес-функція, яка надає можливість користувачу з адміністративним доступом до системи створити звіт з діяльності клініки. Для надання вибірки з наступним редагуванням у обраний формат необхідно ввести початкову і кінцеву дату, за якою будуть обиратися данні.

1.5 Аналіз виявлених проблем діяльності ветеринарних закладів

Проаналізувавши основні джерела інформації, що описують власне ветеринарну діяльність, та точково підступаються до переваг і недоліків віддаленого (онлайн) запису на прийом були відібрані найбільш суттєві проблеми звичайного запису на прийом:

1. Застарілий спосіб зберігання документації, що представлено паперовим документообігом у ветеринарному закладі;
2. Необхідність присутності клієнта у ветеринарному закладі, у процесі стандартного запису на прийом, який зазвичай виконують через клінічну реєстратуру;
3. Велике навантаження на адміністратора(ів) закладу, що змушує власника задуматися про найм нових працівників, що в свою чергу негативно впливає на бюджет компанії;
4. Неможливість клієнтами закладу у будь-який момент переглянути клінічну історію свого улюбленця;
5. Залежність оформлення стандартного запису на прийом від людського фактора настрою адміністратора клініки.

Проблема застарілого способу зберігання документації є дуже обширною, адже вона зачіпає, як і власне клініку, так і природу. Адже, папір це не місце на жорсткому диску комп’ютера, він не береться з «нічого», він потребує попереднього створення, задля чого знищується величезна кількість рослинної екосистеми планети. По-друге, паперова документація, знову ж таки, на відміну від документації електронної потребує місце, багато місця, особливо при розростанні роботи ветеринарного закладу. По-третє, паперова документація схильна до такого фактору – як псування, адже її не можна проста так взяти і перенести з одного дискового накопичувача на інший. Вирішенням даної

проблеми можна назвати перехід з паперового документообігу на електронний, починаючи від реєстрації клієнта у системі і закінчуючи веденням клінічної історії тварин, що лікуються у клініці.

Необхідність присутності клієнта при оформленні запису на прийом – є дуже важливим недоліком у наш час, адже переважна більшість сучасних людей звикла до постійного перебування у онлайні і вже просто не сприймають виконання простих справ за допомогою походів до закладів, чи навіть до магазинів. Клієнти прагнуть свободи, навіть у даному питані, адже оформити запис самому, навіть о 12 годині ночі є набагато кращим варіантом, аніж спочатку приїхати до закладу, після чого дізнатися, що запис на прийом, на бажану дату неможливий. Дана проблема вирішується за допомогою створення повноцінного онлайн-запису на прийом, що надасть більшого задоволення клієнтам. Однак, форма запису на прийом не повинна представлять собою стандартну форму, типу «Ім'я-Проблема», після чого з вами зв'яжуться. Це повинна бути повноцінна форма запису, з можливістю вибору типу послуг на прийом, дати і часу прийому.

Проблема навантаження на адміністратора клініки є стандартною для закладів лікувального типу. Адже, вислуховувати кожен день один і той же текст по сотні раз не є приємним задоволенням. Хоча і система припускає можливість зв'язку адміністратора з клієнтом, однак цей зв'язок по своїй суті є звичайним підтвердженням запису, з можливим легким коригуванням запису. Також, в системі буде присутня можливість користувачем відмовитися від зв'язку з адміністратором, що автоматично підтверджує запис на прийом. Саме тому, для ведення такого типу роботи достатньо одного, край – двох адміністраторів, враховуючи стандартний потік клієнтів. Що дозволяє власнику не задаватися питанням щодо найму нових працівників, а націлити курс ветеринарного закладу на поліпшення задоволення клієнтів від сервісу.

Ведення паперового документації призвело до ще однієї проблеми – а саме неможливість клієнтами ветеринарного закладу переглянути клінічну історію свого домашнього улюбленця. Адже вся інформація зберігається на папері, десь у приміщенні ветеринарного закладу. І щоб отримати інформацію про останнє щепленням необхідно прийти до клініки, що не дуже добре відгукується на задоволенні клієнта від сервісу. Рішенням даного питання є створення у онлайн-форматі, в користувальницькому кабінеті інформацію щодо зареєстрованих

тварин даного клієнта, аби у будь-який час клієнт міг дізнатися всю інформацію про попередні записи на прийом, та головну інформацію про тварину. Система «витягатиме» дані з бази даних. Що унеможливить надання неправильних даних.

Залежність людського фактора було вирішено занести до проблем заданої предметної області. Адже, для оформлення стандартного запису на прийом клієнт повинен віч-на-віч «зіштовхнутися» з адміністратором, тобто людиною яка оформлюватиме запис на прийом. Людина – не машина, їй не властива можливість виконання команд за дорученням і у будь-який час вона може зірватися і нагрубити клієнтові, це і є людським фактором. Систему можна запрограмувати, а людину – ні, як мінімум на даний момент часу це не є можливим. Рішенням даної проблеми є згадувана раніше система онлайн запису на прийом, що мінімізує перетин клієнта з адміністратором. Адже, у цьому випадку всю роботу по оформленні запису на прийом на себе бере запрограмована послідовність дій і клієнт взаємодіє з варіантами, які йому надає система. Саме тому, створювана система не повинна містити помилок.

Аналізуючи все написане вище можна прийти до наступного висновку – розроблювана та досліджувана система інформаційного забезпечення функціонування ветеринарного закладу покликана вирішити проблемні зони застарілого способу запису на прийом у ветеринарному закладі. Система повинна містити функціонал реєстрації користувача, для ведення звітності щодо здоров’я улюбленця. Також, повинна бути присутня можливість онлайн запису на прийом, що вирішує проблеми з навантаженням на адміністратора та клієнта ветеринарної клініки. Можливість ведення документації по прийомам у ветеринарному закладі в електронному вигляді позбавить проблеми з паперовою документацією, та й власне модернізує заклад, що позитивно вплине на оцінку закладу його клієнтами, та на потік нових клієнтів.

1.6 Постановка задачі дослідження

Проаналізувавши сучасний стан підходу до автоматизації функціоналу ветеринарних закладів за необхідне вважаю створення для закладів такого типу інформаційних технологій електронного документообігу у рамках інформаційної системи функціонування ветеринарного закладу.

Для вирішення проблеми застарілого способу зберігання документації, який стисло можна описати, як паперовий документообіг необхідно створити інформаційну технологію електронного документообігу, яка дозволить з легкістю підійти до процесу зберігання даних. Даний функціонал повинен дозволяти з легкістю зберігати та виконувати операції стандартної обробки даних. Для досягнення захищеності даних повинна бути створена система доступу до даних, тобто доступ до даних повинен бути доступний обмеженому колу співробітників. Ключовою особливістю такого підходу слід вважати реалізацію електронних документів замість, використовуваних зазвичай, паперових з метою накопичення, надання та обміну бажаної інформації.

Даний функціонал дозволить «вдихнути нове життя» на звичайний стиль зберігання документів. Враховуючи можливості такого підходу наступною задачею є створення функціоналу для онлайн запису на прийом. Під онлайн записом на прийом слід розуміти введення авторизованим, а тому й зареєстрованим користувачем у систему даних до форми запису на прийом. Дані включають у себе наступні:

- клика тварини;
- порода тварини;
- дата народження;
- фото (за бажанням);
- тип послуги на прийомі;
- дата і час прийому.

Введені дані повинні бути збережені у базу даних, що в майбутньому дозволить мати прямий доступ до них та використовувати у цілях надання користі клієнтам закладу.

Реалізація онлайн запису на прийом відкриває можливість для створення повноцінної картки лікування тварини, яка буде містити повну інформацію про тварину та її господаря, що дозволить ветеринарному закладу відмовитися від ведення паперового обліку клієнтів та тварин. Картка повинна заповнюватися під час оформлення запису на прийом, та після його проведення.

Картка повинна містити наступну інформацію:

- кличка тварини;
- дата народження;
- порода;

- захворювання тварини;
- курс лікування;
- дати прийомів;
- ПІБ господаря;

Інформація повинна бути доступна клієнту у будь-який момент за рахунок використання особового запису у застосунку. Тварини клієнта повинні бути програмно «пов’язані» з ним, аби кожен клієнт бачив інформацію лише про свого улюбленця. За бажанням клієнта картка повинна бути перетворена у «.pdf» формат для наступного зберігання на носії клієнта.

Для автоматизації роботи адміністраторів закладу повинен бути створений інтерфейс для обробки та підтвердження запису на прийом. Що дозволить адміністратору без зайвої писанини вести роботу з обслуговування клієнтів. Функціонал обробки і підтвердження запису на прийом повинен бути інтуїтивно повний і зрозумілий, аби адміністратор без зайвих рухів задовольняв потреби клієнтів та ветеринарного закладу. Доступ до даного функціоналу повинен надаватися лише для адміністраторів закладу, аби уникнути проблем з несанкціонованим доступом до даних користувачів. На виході даний функціонал передбачає підтверджений чи скасований запис на прийом, що відмічається певним атрибутом у базі даних. За бажання клієнта аспект обробки запису з адміністратором можна відхилити, в цьому випадку система повинна автоматично підтвердити запис на прийом.

Для зручності орієнтації клієнта у прийомах необхідно створити функціонал автоматичного оповіщення клієнта про підтвердження чи скасування запису на прийом. Оповіщення клієнта повинно бути представлене у вигляді повідомленням, створеного за допомогою додавання клієнтської інформації до шаблону оповіщень. Даний документ повинен бути надісланий клієнтові відразу після процедури обробки запису на прийом. Електронний документ може виступати підтвердженням клієнта при відвідуванні закладу.

Для ведення звітності роботи закладу необхідно реалізувати функціонал для створення звітів з роботи закладу. Функціонал повинен містити шаблони для генерації добового, тижневого, місячного та річного звіту з діяльності ветеринарного закладу. Доступ до функціоналу створення звітів повинен надаватися адміністраторам закладу. Звіти повинні містити повну інформацію про роботу клініки, включаючи загальну кількість проведених прийомів,

кількість грошей (грошові надходження за надані послуги), що було зароблено, години витрачені на процес прийому. Бажано реалізувати функціонал аналізу даних звітів для виведення успішності роботи закладу та визначення найсприятливіших факторів для роботи ветеринарного закладу.

Для виконання даного функціоналу інформаційна система ветеринарного закладу повинна відповідати певним критеріям, тобто бути реалізована таким чином, який би дозволив дослідження та власне використання бажаного функціоналу. Для чого необхідне виконання певних системних кроків. Тобто, необхідно розробити таку інформаційну систему, яка надаватиме користувачам можливість у форматі «онлайн» оформити запис на прийом до ветеринарного лікаря. Для підвищення ефективності роботи та задоволення користувача планується створити електронне направлення на прийом, яке створюється після підтвердження/скасування запису на прийом. Дане направлення створюватиметься автоматично, без допомоги з боку адміністратора, тобто самостійним чином. Таким чином, ми позбавляємо адміністратора від рутинної роботи з написання подібних листів клієнтам.

Для апробації розробленої ІТ електронного документообігу потрібно розробити клієнтську та серверну частини застосунку. Розроблювані частини застосунку повинні бути створені на сучасних технологіях, що дозволить застосовувати новітні рішення при виникненні проблем. Серверна частина системи повинна бути реалізована відповідно до «REST»-підходу.

2 ОГЛЯД МЕТОДІВ ТА ТЕХНОЛОГІЙ, ЯКІ ЗАСТОСОВУЮТЬСЯ В ПРЕДМЕТНІЙ ОБЛАСТІ

2.1 Технологія електронного документообігу

Документообіг є сукупністю інформації, що створюється і циркулює на підприємстві, або поступає ззовні, чи створюється для оприлюднення поза межами підприємства [14]. Електронний документообіг включає як зовнішні так і внутрішні документи організації. Відмінність полягає в тому, що електронний документообіг передбачає процес створення документів в електронному форматі, тобто документів інформація в яких зафіксована у вигляді електронних даних.

До етапів роботи з електронним документообігом відносять наступні процеси:

- обробка електронних документів;
- створення електронних документів;
- відправка/передача електронних документів;
- отримання/надання факту про отримання електронного документу;
- редагування електронного документу;
- збереження електронних документів;
- видалення електронних документів.

Ключовою ознакою електронного документообігу вважається факт створення електронних документів, замість паперових із метою зберігання, надання та обміну електронної інформації.

Електронний документообіг, у порівнянні зі стандартним паперовим, дозволяє вивести облік та планування дій підприємства на новий рівень. Даний факт суттєво знижує можливі робочі ризики.

Створення шаблонів документів у електронному вигляді суттєво зберігає час створення цих документів, та відкидає аспект створення помилок при ручному оформленні документів. Обмін електронними документами відбувається майже миттєво, за умови стабільного доступу до системи інтернет, що в свою чергу допомагає відмовитись від витрат на папір, друк та використання кур’єрів для доставки.

Важливою перевагою електронного документообігу є той факт, що електронні документи можна створювати, зберігати, редагувати та використовувати в єдиному електронно-інформаційному просторі, що вбереже від аспекту втрати інформації.

Єдиний електронно-інформаційний простір надає можливість регулювати кількості учасників і користувачів системи. Тобто, доступ до даних має лише обмежений круг осіб, що виводить захисту даних на новий рівень.

Для підприємства перехід на електронний документообіг є процесом переведення всіх документопотоків у цифрову форму і автоматизацію процесів обробки документів.

Системи внутрішнього електронного документообігу забезпечують рух електронних документів між підрозділами організації [15]. Підприємства мають таку практику, що всі документи, у особливості фінансові, перед виходом назовні мають бути погоджені з усіма відповідальними особами присутніми на підприємстві. Завдяки можливості налаштувати ланцюг даного погодження електронного документу цей процес можна прискорити в рази. Процес такого прискорення спрощує погодження документів підприємства, що найчастіше потребують уваги.

Зовнішній електронний документообіг, на відміну від внутрішнього надає можливість обміну електронними документами між партнерами та контрагентами підприємства. Тобто, зовнішній електронний документообіг це рух документів спрямований назовні. При зовнішньому електронному документообігу кожен електронний документ перед його відправкою підписується кваліфікованим електронним підписом та шифрується на сертифікат партнера, що надає йому юридичну силу та захист від перехоплення сторонніми особами.

КЕП, або кваліфікований електронний підпис є удосконаленим електронним підписом, який створюється з використанням засобу кваліфікованого електронного підпису і базується на кваліфікованому сертифікаті відкритого ключа. КЕП складається з двох частин – секретного і відкритого ключа, за допомогою яких можна перевірити справжність надання електронного підпису [16].

Для організації внутрішнього документообігу слід зробити наступні кроки:

- визначитися з переліком операцій, які потребують переводу на електронний документообіг;
- створити перелік електронних документів;
- створити внутрішнє положення підприємства про електронний документообіг, що включатиме порядок виправлення помилок електронного документообігу та правила роботи з електронними документами;
- визначитися щодо питань умов визнання електронного документу відправленим.

Із впевненістю можна вважати перехід підприємства на електронний документообіг вигідним і інноваційним кроком у майбутнє [17]. Адже дана можливість суттєво зменшує витрати підприємства за рахунок відмови від канцтоварів та процесів друку і доставки документів. Безперечною перевагою є економія часу працівників підприємства та швидкість обміну електронними документами. Можливість працювати з електронними документами з будь-якої точки світу, за умови отримання до них доступу, є не менш важливим фактором, який вказує на необхідність введення системи електронного документообігу на заміну стандартному паперовому документообігу.

2.2 Технології реалізації клієнтської частини застосунку

Клієнтська частина являє собою графічний інтерфейс застосунку, тобто те що користувач безпосередньо бачить на екрані свого гаджету. Частіше всього клієнтська частина представляє собою набір скриптів, які написано за допомогою мови програмування JavaScript, та придатних до використання в браузері клієнта веб-застосунку.

Клієнтська частина застосунку завантажується на пристрої користувача у вигляді динамічних веб-сторінок. Дані веб-сторінки запускаються на будь-яких гаджетах (пристроях) та операційних системах, в яких наявний інтернет-браузер [18].

Клієнтська частина застосунку, або як її називають у ІТ спільноті «клієнт» відповідає за наступні функції [19]:

- визначення структури веб-сторінки;
- налаштування зовнішнього виду сторінок веб-сторінки;

- реалізація механізму процесу взаємодії користувача, що включає в себе натискання кнопок, роботу різноманітними полями для вводу, тощо.

Логіка клієнтської частини застосунку ґрунтується на використанні DOM, та необхідності надсилати AJAX запити за «вказівкою» користувача [20].

DOM, що розшифровується як об’єктна модель документа, представляє собою загальну структуру веб-сторінки. Робота з ним означає пошук, додавання, зміну та різноманітні операції з переміщення HTML-тегів.

AJAX, у свою чергу, представляє собою технологію, яка дозволяє виконувати асинхронні запити до веб-серверу, та таким чином обмінюватися даними [21]. Однак, є деяка проблема, адже клієнтська та серверна частини застосунку написані на різних мовах програмування, що не дозволяє за допомогою однієї команди обмінюватися даними. Для цього й виникає необхідність обмінюватися не простими даними, а перетвореними у JSON формат.

JSON – позначення об’єктів JavaScript – є універсальним форматом, який необхідний для обміну даними між клієнтською і серверною частиною застосунку. JSON файл представляє собою звичайний «рядок», тобто код перетворений до рядкового типу, який може бути використаний у будь-якій мові програмування. Процес приведення списку чи об’єкту даних до формату JSON називається серіалізацією. Зворотній об’єкт прийнято називати десеріалізацією.

2.2.1 Огляд технології HTML

Макет і вміст веб-сторінки визначається за допомогою HTML, на даний момент HTML5. Він і виступає головною мовою для опису графічного інтерфейсу веб-застосунку [22].

HTML являє собою мову гіпертекстової розмітки і використовується для того аби повідомляти браузеру користувача яким чином відображати веб-сторінки, якій той у свою чергу відвідує. Процес даного повідомлення має наступний формат – коли користувач заходить на бажану веб-сторінку, то браузер підвантажує необхідний HTML-файл, що містить структуру та контент веб-сторінки. HTML ніби вибудовує візуальний фундамент сайту, однак не запускає його самостійно, це лише структура. Він вказує порядок розташування елементів, їх базовий дизайн та вкладеність тегів сторінки. HTML-файл, частіше за все містить і інформацію звідки брати стилі для елементів сторінки, тобто їх

зовнішній вигляд, та скрипти для них, які визначають базовий функціонал веб-елементів застосунку [23].

HTML складається з ряду елементів, які розробник використовує, аби вкладати різні частини контенту сторінки, щоб змусити їх відображатись так, як необхідно йому. Цей контент можна зробити посиланням на якусь сторінку, анімацією, чи звичайним текстом, написаним з різним форматом, починаючи від жирного і закінчуючи шрифтом типу курсив. Дані елементи являють собою теги, тобто команди, які вказують браузеру, як відображати інформацію, яку вони зберігають.

Для прикладу елементу можна навести текстовий абзац, що зображено нижче, у лістингу 2.1.

Лістинг 2.1 – Програмний код HTML-елементу, типу текстового абзацу

```
<p>Розробка та дослідження інформаційного забезпечення функціонування ветеринарного закладу</p>
```

Як можна побачити вище, стандартний HTML-елемент складається з 4 елементів:

- відкриваючий тег; представляє собою ім’я тегу елементу, що укладено у кутові дужки які відкриваються і закриваються; відкриваючий тег позначає собою місце у коді, де елемент починає свою роботу;
- контент HTML-елементу, який у даному випадку представлений звичайним текстом;
- закриваючий тег; являє собою те ж і саме, що й відкриваючий тег, за виключенням того факту, що він містить косий символ перед ім’ям елементу; закриваючий елемент у коді позначає місце, де елемент закривається, або закінчує роботу; відсутність закриваючого тега відносять до найрозповсюдженіших помилок при роботі початківців, адже відсутність одного такого тегу призводить до краху всього коду;
- елемент; включає у себе всі перелічені вище частини HTML-елементу.

HTML – це каркас сайту, який надає можливість браузеру відмалювати веб-сторінку, та дає доступ до посилань на CSS та JavaScript. На цьому функціональні його функціональні можливості завершуються, адже це не мова програмування, а лише формат тексту.

2.2.2 Огляд технології CSS

CSS, який прийнято розшифровувати як «каскадні таблиці стилів» використовується для визначення зовнішнього вигляду веб-сторінки. Це мова, яка дозволяє описати стилі тегів, тобто елементів, які було задано у HTML, тобто код який розробник використовує для стилізації веб-сторінки [24].

По своїй «натурі» CSS, як і HTML не прийнято називати мовою програмування, однак це й не мова розмітки, це мова таблиць стилів. Це означає, що використання CSS, дозволяє застосовувати стилі до вибіркових HTML-елементів. Наприклад, аби надати жовтого кольору всім текстовим елементам на сторінці необхідно написати код, наведений у лістингу 2.2.

Лістинг 2.2 – Таблиця стилів для текстового HTML-елементу

```
p {
color: yellow;
}
```

Структура набору правил CSS має наступні елементи:

- селектор; являє собою ім’я HTML-документу на початку набору правил. Він вибирає елементи для застосування певного стилю, у нашому випадку текстовий елемент «p»;
- оголошення; правило, яке вказує на властивості, які треба змінити у обраного елементу, у нашому випадку «color: yellow», тобто зміна кольору елементу на жовтий;
- властивості; способи, якими розробник може стилізувати певний HTML-елемент, у даному випадку «color» являє собою властивість для текстових елементів;
- значення властивостей; значення властивості записується праворуч від властивості, воно і обирає одну з можливих ознак для цієї властивості; беручи до уваги наш приклад, «yellow» - це один із безлічі варіантів для властивсості «color».

CSS призначений для відокремлення того, що визначає загальний вид сторінки, від її змісту [25]. Можна створити веб-сторінку і без файлу типу CSS, однак можливість читання даної веб-сторінки розробником зведеться до мінімальних значень. Тому прийнято, що HTML описує черговість об’єктів, віддаючи перевагу опису стилю тегів CSS.

2.2.3 Огляд технології JavaScript

Для реалізації механізму взаємодії з користувачем необхідний JavaScript. JavaScript – це мова програмування, яка додає інтерактивності на статичний веб сайт, наприклад відгук на введення інформації у поля форми чи натискання користувачем певних кнопок.

JavaScript, а надалі JS вважається повноцінною динамічною мовою програмування, яка застосовується до структури HTML, та надає можливості забезпечення динамічної інтерактиності на веб-сайтах [26].

JS, як інструмент є доволі гнучким і може використовуватися у багатьох аспектах програмування, починаючи від створення ігор і завершуючи створенням застосунків із використанням баз даних.

JS є дуже гнучким і компактним, що дозволило розробникам, поціновувачам даної мови, створити величезну кількість інструментів, які працюють поверх основного двигуна JS. Вони допомагають відкрити доступ до можливостей, які не були доступні на нативному JS, до них можна віднести наступні [27]:

- API, тобто програмні інтерфейси застосунків, які вбудовуються у браузер та дозволяють користувачам проводити складні операціїї, такі як динамічне створення HTML та стилів, робота з веб-камерою користувача, тощо;
- API сторонніх застосунків, що дозволяють користуватися стороннім функціоналом не вдаючись до важких маніпуляцій;
- фреймворки та бібліотеки, які дозволяють розширити функціонал застосунків і можливостей розробника.

Дивлячись на написане вище можна подумати, що JS є чимось дуже легким, однак все не так, як здається. Освоїти JS набагато важче, аніж HTML та CSS, адже це повноцінна мова програмування, що виводить її на новий рівень осмислення, порівняно з пригаданими технологіями. І побачити це помжна з прикладу оформлення коду JS файлу, який наведено у лістингу 5.3.

Лістинг 5.3 – Приклад оформлення JS-коду

```
let myText = document.querySelector(“p”);
myText.textContent = “Baku Andrey”;
```

Цей код виконує просте перетворення тексту елемента «p» на сторінці, на задану умовою коду фразу. За допомогою функції «querySelector()» ми створили посилання на текстовий елемент «p» у нашому HTML, тобто отримали власне сам елемент, його значення. Отримане посилання на елемент було збережено до змінної «myText». Змінна у JS являє собою контейнер, який може зберігати будь-яке значення. Даний процес дуже схожий на вибір елемента у CSS, адже для роботи з елементом потрібно попередньо його вибрати.

Після чого, для властивості обранного елемента, яке визначає його текстове наповнення було введено нове тестове значення.

JavaScript є «найкращим другом» для розробки з використанням HTML та CSS. HTML необхідний для визначення розмітки сайту, CSS потрібний для надання зовнішнього вигляду веб-сторінки, а JS відповідає за надання інтерактивності всього створеного на HTML та CSS, тобто зі статики переводить у динаміку. За допомогою JS розробник визначає, що і як на веб-сторінці буде реагувати на певні дії користувача. На даний момент JS є єдиною мовою програмування для браузерів.

2.2.4 Огляд технології React

React є однією з найпопулярніших JavaScript бібліотек. Він виступає інструментом для створення користувальницьких інтерфейсів. Його задачею є забезпечення виводу на екран того, що можна буде побачити на веб-сторінках [28].

Головною фішкою React є розбиття інтерфейсу веб-сторінки на так звані фрагменти, які прийнято називати компоненти [29]. Використання компонентів значно полегшує створення ємнісних інтерфейсів. Не меншою аніж компоненти можна назвати особливість у вигляді станів, які зберігають всю інформацію про елемент, у тому числі й інформацію про відображення елементу.

Компонетом у React можна назвати ділянку коду, яка представляє собою певний «шматок» веб-сторінки, нехай то буде футер, чи головне меню застосунку.

Для написання React застосунків використовується індивідуальна для нього мова програмування – JSX – яку можна інтерпритувати, як HTML, який працює всередині JavaScript, тобто отримуєму універсальну мову для написання веб-сторінок [30]. Приклад написання React коду, використовуючи JSX нотацію, наведено у лістингу 5.4.

Лістинг 5.4 – Приклад написання коду React застосунку

```
function getUserGreeting(userData) {
if(user) {
return <p>Hello, {userData.name}!</p>
}
return <p>Hello, unknown user.</p>
}
```

З написаного коду можна побачити те саме об’єднання JavaScript та HTML. Адже, замість стандартних функцій для пошуку елементу у HTML, ми маємо чистий HTML у стандартному JavaScript. Даний код зображує функцію, яка приймає дані користувача, у разі якщо дані наявні, то виводить на екран текстовий блок із текстом-привітанням, у зворотньому випадку на екрані з’являється текстовий блок з привітанням незнайомого користувача.

React має наступні переваги перед стандартним JavaScript:

- дуже зручний у читанні та розумінні розробником, за рахунок чого на підтримку та налагодження коду потрібно витрачати менше часу і ресурсів;
- реалізована система компонентів, які можуть використовуватись у різних умовах, за рахунок чого досягається неймовірна економія часу і чистота коду;
- очевидність помилок у коді, за рахунок того, що кожна компонента залежить лише від свого стану, тобто помилки можна відслідковувати локально.

В підсумку, маємо те, що React допомагає суттєво заощадити час розробки застосунку, робить код більш зрозумілим і структурованим, за рахунок використання магії компонент. Все це допомагає заощадити на вартості розробки, підтримки та налагодження застосунку, а що найголовніше робить сам застосунок неймовірно швидким. React – це спосіб для зручного представлення коду HTML і JavaScript, з можливістю зробити його повторюваним і наочним.

2.3 Технології реалізації серверної частини застосунку

Серверна частина – це основна частина застосунку, яка містить в собі всі обчислення [31]. Обчислення і різноманітні функції виконуються на основі інтернет-запитів користувачів застосунку, які надсилаються через браузер.

Головною особливістю роботи сервера є той факт, що одночасний доступ до серверної частини застосунку може мати велика кількість користувачів. Однак написання застосунку використовуючи лише серверну частину не є доцільним, адже на виході отримується веб-застосунок, який буде перезавантажуватися майже від будь-якої дії на сторінці, що не є гарною практикою.

Для розробки серверної частини веб-застосунку прийнято залучати розробників, знайомих з технологіями PHP, Python, Java і т.д.

Програмування серверної частини застосунку використовується для динамічного відображення даних лише при їх необхідності. Дані беруться з бази даних, яка зберігається на сервері і надсилає дані користувачу для відображення через деякий код.

Найбільша користь серверного програмування в тому, що воно дозволяє формувати індивідуальний контент веб-сторінки під кожного конкретного користувача. Динамічні сайти мають можливість виділяти певний контент, який прийнято вважати актуальнішим для клієнта, в залежності від його дій. Даний процес допомагає спростити використання веб-сторінок за рахунок збереження особистої інформації, наприклад можливість повторного використання збережених кредитних даних користувача, для оптимізації і швидкості наступного платежу.

Даний потенціал надає можливість взаємодіяти з користувачем сайту, надсилаючи повідомлення та ймовірні оновлення за допомогою електронної пошти, та інших веб-каналів. Все це дозволяє в значній мірі глибше взаємодіяти з користувачами застосунку.

2.3.1 Огляд технології Python

Python – це мова програмування загального призначення, яка широко розповсюджена при розробці інтернет-застосунків, розробці загального програмного забезпечення, науці про дані та створенні штучного інтелекту [32]. Python вибирають через його ефективність та відносну простоту при вивченні [33]. Ця універсальність, паралельно зі зручністю для початківців, зробили його одним з найчастіше використовуваних мов для програмування.

Загалом, роботу програми на Python, можна описати за допомогою декількох кроків:

- програма читається парсером, тобто аналізатором синтаксису, в результаті чого отримується набір лексем для майбутньої обробки;

- після чого парсер, за допомогою інструкції виконує генерацію структури і формування синтаксичного розбіру (AST);
- надалі, компілятор перетворює AST у байт-код і видає його на виконання інтерпритатору;
- наприкінці інтерпритатор коду построково виконує всі надані йому інструкції.

Програму у Python прийнято називати скрипт або сценарій.

Python часто використовується для веб розробки, а саме розробки внутрішньої частини веб-застосунку, тобто тих частин, яких користувач не бачить [34]. Роль Python у веб розробці дуже обширна, починаючи з відправлення даних на сервери, та назад, обробки отриманих даних і закінчуючи маршрутизацією URL адреси та забезпечення безпеки системи. Для веб-розробки з Python найчастіше використовуються два фреймворки – Django та Flask.

Іноді, при написанні Python застосунків можуть виникати труднощі, що пов’язані з застосуванням до продукту різних версій бібліотек. Версії яких можуть відрізнятися і для роботи однієї необхідна більш стара версія іншої. Для вирішення проблем такого типу існують віртуальні середовища або оточення. Кожна з віртуальних середовищ дозволяє запускати свій застосунок з різним набором бібліотек. Та зміна версій цих бібліотек ніяк не впливає на інші застосунки.

У стандартну бібліотеку Python входить модуль venv, який і формує віртуальне середовище [35].

Плюсами Python слід вважати його простоту, велику кількість вбудованих і зовнішніх бібліотек та фреймворків, які допомагають досягти будь-якої мети. Відкритий вихідний код, велике ком’юніті розробників, можливість прочитати та зрозуміти код дуже допомагає зменшувати час на розробку коду.

2.3.2 Огляд технології Django

Django є високорівневим Python веб-фреймворком, який дозволяє швидко створювати захищені та підтримувані веб-сайти [36]. Django створений для того, аби взяти на себе більшу частину клопоту веб-розробки, тому його використання дозволяє зосередитися на використанні його можливостей, а не на «винаходженні велосипеду».

Django слідує філософії «Все включено» і надає все, що необхідно веб-розробнику прямо з коробки [37]. Адже, все. Що потрібно при розробці веб-продукту бездоганно працює у зв'язці, відповідає принципам проектування та має обширну документальну базу.

Під ефективним Django слід розуміти його використання таким чином, щоб написаний код був зв'язковим, тестованим та масштабованим.

Під зв'язковим кодом треба розуміти код, який зосереджений на виконанні тільки однієї речі. Що означає те, що при написанні функції чи методу необхідно націлюватися на виконання чогось одного, а не намагатися впхнути все у один шматок коду.

Зв'язковий код безпосередньо впливає і на розуміння тестованого коду, адже код, який одночасно виконує багато речей не є правильним для тестування. Тобто, під тестовим кодом слід розуміти такий код, який дозволяє писати для нього прості і ефективні тести. Тобто, код в якому можна легко знайти помилку.

Поняття коду, що масштабується витікає з його назви, тобто код повинен без величезних зусиль збільшуватись із часом за рахунок додавання до проекту нових інженерів. Тобто, маємо наступний висновок – добре протестовані застосунки легші для розуміння, що передбачає можливість легкого процесу масштабування.

На традиційному веб-сайті, веб-застосунок чекає на HTTP-запити від браузера. Коли запит отримано, програма розроблює те, що необхідно на основі URL-адреси, тобто даних POST або GET запитів. Залежно від того, що необхідно зробити, він може читати або записувати інформацію з бази даних, чи виконувати інші завдання, необхідні для задоволення запиту. Після чого програма повертає відповідь браузеру, часто динамічно створюючи HTML-сторінку для відображення, вставляючи отримані з бази даних дані в HTML-шаблон.

Принцип роботи Django фреймворка можна описати наступним чином:

- першим чином запит від користувача попадає в роутер, завданням якого є вирішити яку функцію для обробки запиту необхідно викликати; вирішення відбувається за рахунок списку правил, типу «певний урл – певна функція»;
- функція, яку викликає роутер можна назвати view і вона може містити будь-яку бізнес логіку, однак найчастіше це чи надсилання даних з БД на користувальницьку частину, чи протилежна дія, у вигляді збереження даних до БД;

- отримані з БД дані готуються до відправки до користувальницької частини наступним чином – вони можуть бути підставлені у спеціальний шаблон, для наступного відправлення у вигляді HTML-файлу; однак у випадку односторінкового застосунку це відбувається лише один раз, у момент початкової генерації HTML-сторінки, після чого відбувається серіалізація даних та її відправка у JSON-форматі.

Django – набір компонентів, що допомагає значно полегшити процес розробки веб-застосунків за рахунок дотримання філософії ефективного коду.

2.4 Вибір середовища розробки веб-застосунку клініки

Головну частину робочого часу розробники проводять у редакторах коду. Це означає, що комфортність та зручність робочого середовища дуже важко переоцінити. Так, звичайно, можна писати код у стандартному текстовому редакторі й зберігати його у вигляді «.html» форматі, однак це не зрівняється із просунутим редактором коду. Особливо, при орієнтуванні редактору на певну мову [37].

Із присутніх середовищ розробки Python слід виділити наступні:

- Python IDLE;
- Visual Studio Code;
- PyCharm;
- Atom;
- VIM.

Python IDLE є редактором, що поставляється разом із Python. Він придатний для базового і спрощеного режиму програмування на Python. Він підходить для початкового програмування, однак у ньому присутні великі можливості, по типу підсвічування синтаксису та базового вбудованого відладчику. Однак, можливість його використання у складних проектах є дуже сумнівною.

Visual Studio Code зарекомендував себе як безкоштовний редактор коду для всіх систем. До його можливостей слід внести наступні – налагодження та підсвічування коду, можливість рефакторингу коду та інтелектуального завершення коду. Присутня можливість інтеграції з Git. Visual Studio Code підтримує багато мов програмування, включаючи різноманітні фреймворки та

бібліотеки для мов. Для початку роботи з Python, слід буде встановити декілька базових пакетів, однак цей процес важко назвати складним. Несумнівним плюсом редактору є його постійна підтримка та оновлення. Visual Studio Code вважається одним із найкращих редакторів коду не тільки для Python, а й для багатьох інших мов програмування. Із мінусів слід визнати, що вбудований термінал не завжди працює так, як розробник очікує.

PyCharm вважається інтегрованим середовищем розробки, орієнтованим спеціально для Python. Так як редактор орієнтований на Python він має доволі широкий набір можливостей, до яких можна віднести автозавершення та інспекцію коду, підсвічування помилок, наявну систему контролю версій та рефакторинг. PyCharm доступний для всіх операційних систем. Головним недоліком IDE слід вважати його ненажерливість по відношенню до ресурсів комп'ютеру, тобто маючи слабку систему не є раціональним рішенням його запускати.

Atom є дуже схожим на Sublime редактор коду. Редактор підтримує широке коло налаштувань і велику кількість необхідних для роботи пакетів. Із плюсів можна виділити легкість роботи та підтримку Python після установки додаткових пакетів. Враховуючи його направленість, його не слід розглядати для роботи з великим проектом.

Vim є текстовим редактором за замовчуванням для систем MacOS та UNIX. Його люблять за розвинні обчислювальні можливості та компактне середовище розробки. Для початківців Vim буде дуже поганим вибором, адже він потребує попереднього вивчення, яке не є дуже простим. Однак, на виході можна отримати доволі функціональний і продуктивний редактор.

Опираючись на інформацію наведену вище, було досягнуто рішення щодо використання у якості головного середовища для розробки середовище Visual Studio Code, адже воно є кросплатформеним, тобто доступне на всіх ОС. Visual Studio Code не є дуже вимогливим до системи комп'ютера, тобто він буде працювати однаково добре, як на старих, так і на сучасних системах. Проблема терміналу не є дуже глобальною, адже вирішується стандартним перезавантаженням терміналу.

2.5 Огляд технологій взаємодії клієнта та серверу

Процес взаємодії клієнта та серверу відбувається за участі протоколу HTTP [39]. HTTP визначають як протокол, що служить для отримання різних ресурсів, як наприклад HTML документів. Даний протокол є основою обміну даних в інтернеті. HTTP є протоколом клієнт-серверної взаємодії, що означає ініціювання запитів на сервер самим одержувачем, яким частіше за все виступає браузер. Отриманий документ може складатись з різних підчастин, наприклад з окремо виділеного тексту, опису структури документа, скриптів, тощо. З цього можна дійти висновку, що основою протоколу є запит від клієнта до серверу і власне відповідь від серверу.

Для запитів, зазвичай використовують методи типу GET, якщо необхідно отримати дані та POST, якщо є необхідність форматувати дані. Додатково у запиті вказується домен сайту, що є хостом, тіло запиту, яке притаманне POST запитам та інша технічна інформація.

У сучасних веб-застосунках використовується протокол HTTPS, який є покращеною версією HTTP, що підтримує шифрування SSL/TLS. Використання технології шифрування вважається аспектом «гарного тону» при розробці веб застосунків.

Django у даній взаємодії виконує наступні процеси:

- при надходженні запиту з браузеру на сервер, він не одразу потрапляє у Django. Початковим етапом є його обробка веб-сервером Nginx. При необхідності статичного файлу, по типу картинки, Nginx самотужки надсилає його у відповіді клієнту, однак якщо запит потребує не статики, то він передається від Nginx до Django;
- однак, Nginx не вміє проводити таку процедуру, тому використовується сервер застосунку, як приклад таких застосунків можна навести uWSGI чи Gunicorn. Саме вони мають функціонал для передачі запиту у Django;
- після обробки запиту Django, він повертає відповідь у форматі HTML сторінки чи даних, обов’язковим є присутність у відповіді коду відповіді. Якщо все добре, то код відповіді – 200; якщо щось трапилося і сторінка не знайдена, то код – 404; якщо помилка сталась на сервері, то надсилається код 500; Ці коди можна назвати найчастішими у роботі веб-застосунку.

До взаємодії між сервером і клієнтом можна віднести ще одну технологію – кешування. Під даним терміном прийнято розуміти концепцію у розробці застосунків, коли замість того, аби кожен раз діставати з БД дані, що часто використовуються, вдаються до зберігання їх у швидко доступному місці.

2.6 Вибір системи управління базами даних проектованого застосунку

Дані застосунку прийнято зберігати у БД. Частіше за все використовуються реляційні бази даних, які містять таблиці із заздалегідь створеними атрибутами, які дозволяють реалізувати зв’язки між даними таблицями.

База даних – є впорядкованим набором структурованих даних, які зберігаються в електронному вигляді [40]. Для управління базами даних були створені системи управління базами даних [41].

У сучасних базах даних дані прийнято зберігати у вигляді рядків і стовпців, які й формують таблицю. Цими даними можна з легкістю оперувати, тобто змінювати, оновлювати чи впорядковувати. У базах даних для виконання запитів використовується мова структурованих записів, або ж SQL.

У світі існує дуже багато типів баз даних. Кожен з яких підходить для тієї, чи іншої задачі. Далі буде наведено список типів БД, які використовуються найчастіше:

- реляційні бази даних, які організовані у вигляді таблиць, що складаються з рядків та стовпців. Даний тип БД забезпечує швидкий та ефективний доступ до структурованих даних;
- об’єктно орієнтовані бази даних, що представлені у вигляді об’єктів, як у об’єктно орієнтованому програмуванні;
- розподілені бази даних, яка складається з двох чи більше частин, які розташовані одночасно на декількох серверах. Дана можливість дозволяє зберігатися базі даних на декількох комп’ютерах для досягнення додаткового рівня захищеності даних;
- сховища даних, що є нестандартним типом бази даних, яке виступає у вигляді централізованого репозиторію для даних. Такий тип забезпечує дуже швидке виконання запитів процесів аналізу даних;

- NoSQL DB, або ж нереляційна БД, призначена для зберігання і обробки неструктурованих або слабоструктурованих даних. Популярність даного типу зростає паралельно з популярністю веб-застосунків.

На даному етапі розвитку СУБД рейтинг популярності очолюють такі СУБД, як MySQL, Oracle, Microsoft SQL Server та PostgreSQL.

MySQL прийнято вважати однією з найпоширеніших СУБД. Вона відноситься до реляційних баз даних з відкритим вихідним кодом. Плюсами даного підходу слід вважати швидкість та гнучкість, що забезпечено підтримкою великої кількості різноманітних типів таблиць. Окрім цього MySQL це безкоштовна система. У сукупності, ці чинники дозволяють використовувати MySQL у системах будь-якого типу, від малих і до величезних.

Microsoft SQL Server, як можна зрозуміти з назви фірмова СУБД від Microsoft. Найкраще розкривається в операційних системах сімейства Windows, однак присутня можливість роботи і з Linux. Система Microsoft SQL Server дозволяє синхронізуватися з продуктами компанії Microsoft, що забезпечує надійний захист даних та простий інтерфейс. Однак, за даний функціонал необхідно платити, як у економічному плані, так і в плані підвищеного споживання ресурсів комп’ютера. За свою можливість використовуватися у ієрархії Microsoft використовується багатьма компаніями.

PostgreSQL можна назвати популярною системою і цю популярність викликає не те, що вона безкоштовна. Вона універсальна, тобто підходить для роботи з більшістю платформ. PostgreSQL можна назвати об’єктно-реляційною СУБД, що дає їй деякі переваги, у порівнянні з реляційними базами даних.

Oracle з першої версії зарекомендувала себе, як функціональна і практична СУБД. Вона постійно розвивається і дороблюється, що дозволяє розширювати необхідний функціонал. Суттєвим мінусом Oracle можна назвати дуже велику вартість ліцензії, тому у більшості своїй дана СУБД використовується лише великими компаніями, які можуть собі дозволити оплачувати ліцензію.

Враховуючи все написане вище, для проектованого застосунку буде застосовано СУБД MySQL. Швидкість та гнучкість БД по мірі необхідності, ось чого потребує веб-застосунок.

2.7 Висновки до технологій, що застосовуються при розробці

Головними технологіями, що будуть використовуватися при розробці клієнтської частини є базові HTML та CSS, без яких не можна представити веб-застосунок. Однак, обидві технології потрібні тільки для надання зовнішнього вигляду. Оживляти функціонал веб-сторінок покликаний JavaScript, на пару з його фреймворком React. Вони і представлятимуть функціонал для клієнтської частини застосунку.

Серверну частину застосунку буде написано на Python мові програмування, разом з веб-фреймворком Django. Ці технології дозволять у гнучкому форматі створити сервер, та функціонал на який орієнтована система.

Іншу частину серверної частини, а саме базу даних представляє реляційний засіб управління базами даних MySQL. Він поєднує в собі багато плюсів, починаючи з безкоштовного ПО, гнучкості роботи, неймовірного інтерфейсу користувача, який представлено WorkBench, що й виводить його на перший план з багатьох СУБД.

Середовищем розробки веб-застосунку було обрано дуже розповсюджений редактор коду Visual Studio Code, який уміщує в себе легкість самого застосунку і можливість потужної розробки у ньому. При необхідності середовище розробки може бути змінено на PyCharm. До зміни може привести розмірність застосунку, так як при великому обсязі папок проекту Visual Studio Code на відміну від повнофункціональної IDE може наробити проблем.

Взаємодія клієнта та сервера буде досягнута за допомогою використання технології HTTPS, яка по своїй суті є звичайним HTTP з підтримкою шифрування, що надасть можливості додаткового захисту для даних застосунку.

3 РОЗРОБКА ВИМОГ ДО СТВОРЮВАНОЇ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

3.1 Розробка системних вимог до створюваної інформаційної технології електронного документообігу ветеринарного закладу

Для створення інформаційної технології електронного документообігу ветеринарного закладу використовується операційна система сімейства Microsoft Windows, а саме Windows 10. Дана система була обрана через специфіку пристрою на якому проводиться створення інформаційної технології.

Однак, приймаючи до факту те, що під інформаційною технологією розуміється створення веб-додатку, тобто додаток, який працює у браузері та обмеження на операційну систему при її використанні відсутні. Користуватися веб-додатком можна на будь-якому комп’ютерному пристрої, який надає доступ до мережі інтернет.

Для реалізації клієнтської частини використовувався інструментарій JavaScript. А саме, бібліотека React для створення користувальницьких інтерфейсів. Разом із бібліотекою React були використані додаткові бібліотеки, які необхідні для коректної її роботи. Однією з таких бібліотек є Redux, яка необхідна для управління станом клієнтської сторони веб-додатку. Для зв’язки даних бібліотек використовується бібліотека React-Redux, яка виступає своєрідним контролером дій, що відбуваються у клієнті.

Для реалізації серверної частини додатку було обрано мову програмування Python, а саме фреймворк Django. Першим етапом роботи якого є отримання запиту від користувача, що потрапляє у роутер, який і вирішує, яку функцію для обробки даних необхідно викликати. Функція, яка викликається роутером є в’ю функцією, тобто вона утримує будь-яку бізнес-логіку, яка частіше за все є роботою з базою даних.

Дані веб-додатку зберігаються у вигляді бази даних типу MySQL. Для цього використовується реляційна база даних, що означає заздалегідь завдані колонки і таблиці даної БД зв’язані між собою. Дані в БД можна створювати, читати, змінювати та видаляти. Для позначення даних взаємодій можна зустріти абревіатуру CRUD (Create Read Update Delete). Для запиту даних у БД використовується спеціальна мова SQL (structured query language). Для

комфортної роботи з MySQL використано його внутрішній інструмент – Workbench, який надає зручний інтерфейс для взаємодії з БД.

Для зберігання електронної документації створена спеціальна папка з обмеженим доступом. Дана папка містить каталогізацію, тобто підпапки, кожна з яких відповідає за збереження власної унікальної документації, починаючи з підтверджень записів на прийом і завершуючи звітами роботи ветеринарного закладу.

3.2 Опис архітектури розроблюваної системи

При розробці даної інформаційної технології на базі інформаційної системи ветеринарного закладу використовувалася клієнт-серверна архітектура веб-додатків. Дана архітектура містить два основних компоненти – клієнт та сервер, до чого можна дійти з її назви. Ланцюжок роботи архітектури зображено на рисунку 3.1.

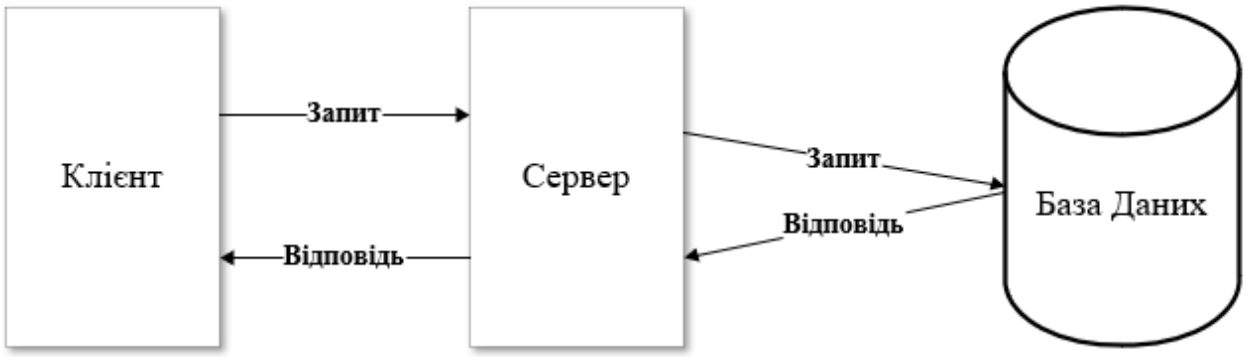

Рисунок 3.1 – Алгоритм функціонування архітектури додатку

Роль клієнта відіграє браузер, що відображає користувальницький інтерфейс веб-додатку. Клієнт необхідний для взаємодії звичайного користувача системи із додатком, тобто роль всеможливих форм та кнопок для взаємодії з ними лежить на клієнтові.

Сервер виступає своєрідним посередником між базою даних та клієнтською частиною. Його роль – отримання запитів від клієнта, створення відповіді та надсилання відповіді на клієнтську частину додатку.

База даних є сховищем системи, тобто у ній зберігається головна інформація ветеринарного закладу, яка необхідна як для роботи додатку, так і для створення різних електронних документів та всеможливих звітів.

Взаємодію системи можна описати у вигляді чотирьох-рівневої моделі архітектури «клієнт-сервер», яка містить чотири компоненти – клієнта, сервер застосунків, базу даних та інструментарій для ведення електронного документообігу. База даних напряму взаємодіє з сервером застосунків, який у свою чергу взаємодіє з клієнтом. Обидва ж, з вищевказаних, напряму взаємодіють з системою електронного документообігу, тобто електронні документи можна робити з даних, що знаходяться на сервері, та тих, які тільки введено на клієнті. Взаємодія між елементами системи зображено на рис.3.2.

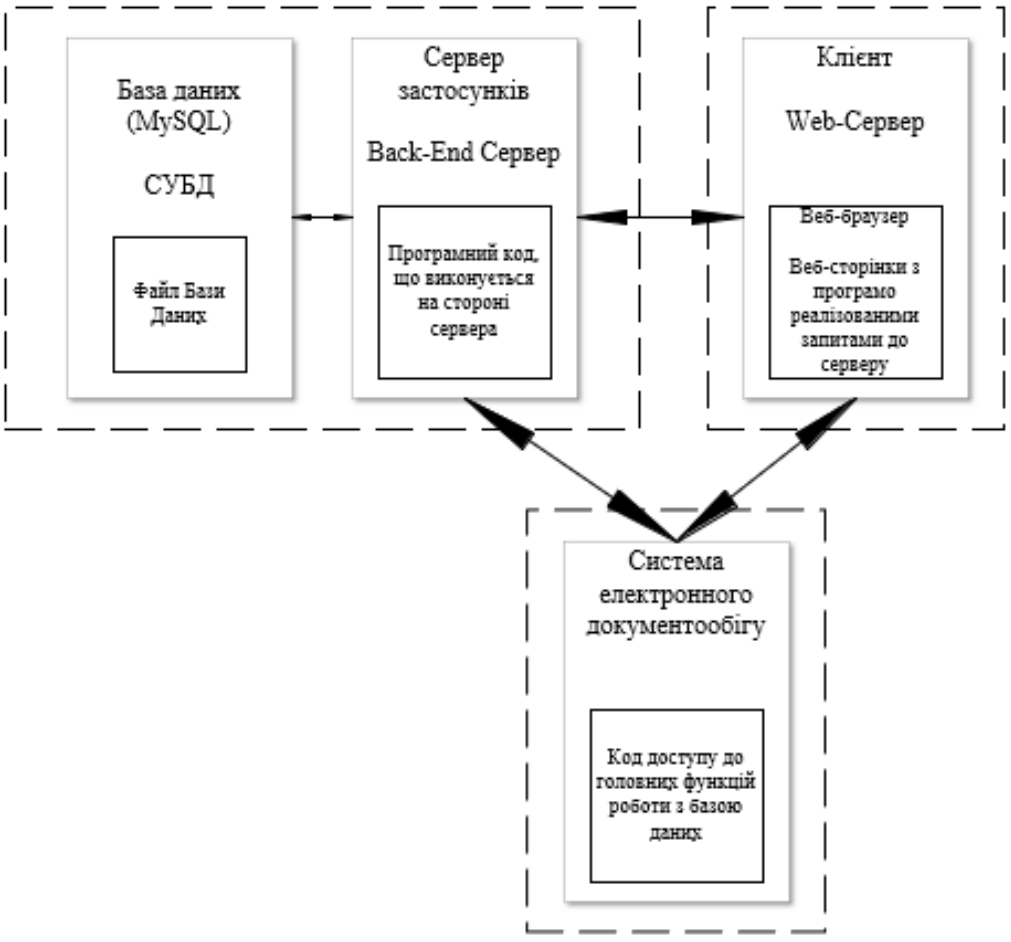

Рисунок 3.2 – Взаємодія між елементами інформаційної системи

Виходячи з вищесказаного про архітектуру системи можна описати принцип роботи електронного документообігу в системі. Вхідні дані від клієнта, що надходять до системи із форм для реєстрації та оформлення запису на прийом

потрапляють на сервер після чого зберігаються у базі даних. Після проведення прийому лікар вносить після-прийомні дані у необхідні атрибути бази даних. Для створення електронних документів необхідні збережені дані вибираються з бази даних, після чого проводиться їх розміщення у шаблонні варіанти документів. Після чого електронні документи, чи їх електронне представлення надсилаються/надаються клієнтам у електронному форматі. Схема електронного документообігу даних у ветеринарному закладі наведена на рис.3.3.

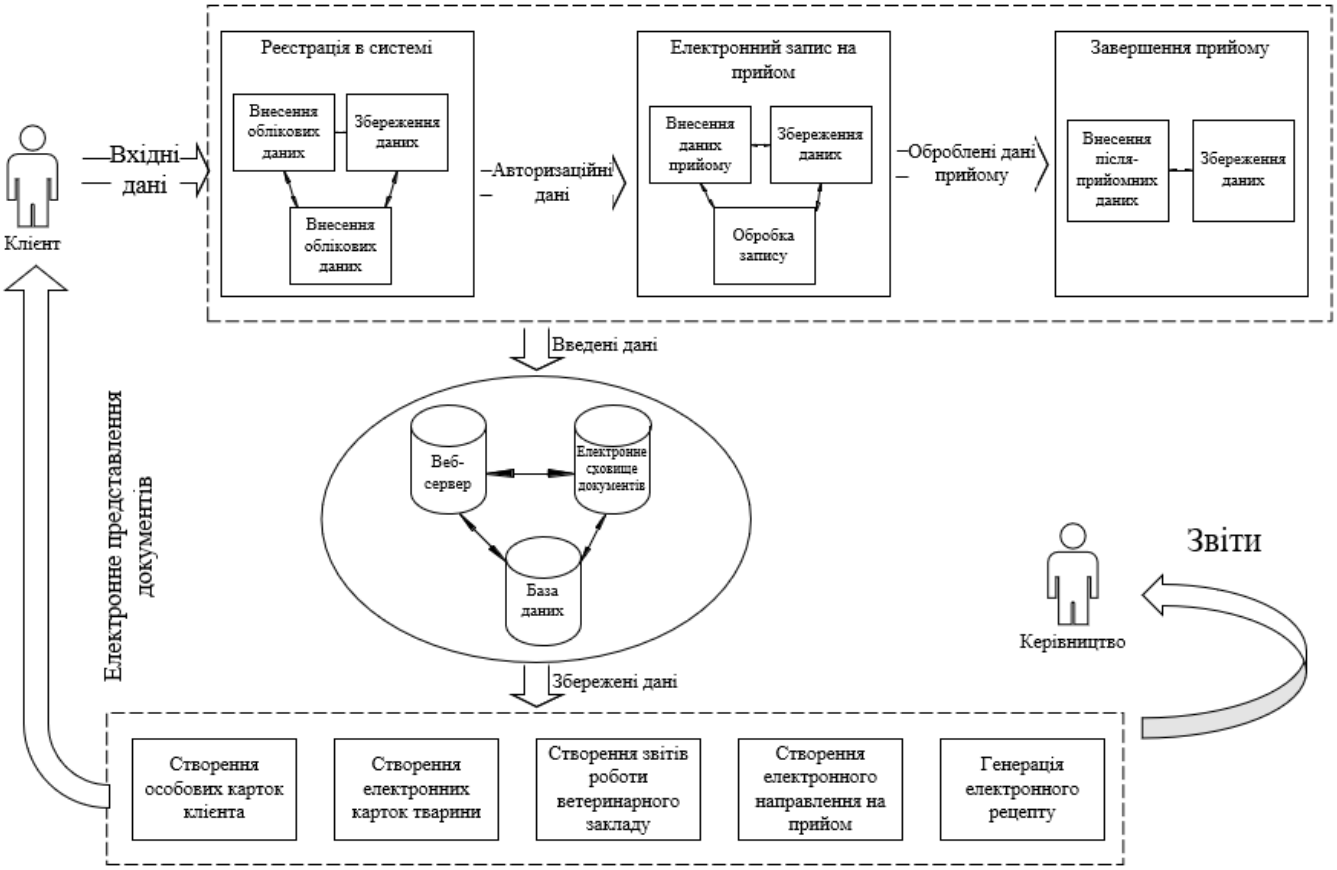

Рисунок 3.3 – Схема електронного документообігу у ветеринарному закладі

3.3 Розробка функціональних вимог до інформаційної технології електронного документообігу ветеринарного закладу

Для розробки функціональних вимог до створюваної інформаційної технології електронного документообігу ветеринарного закладу було проведено процедуру функціонального моделювання з використанням стандарту IDEF0.

На рисунку 3.4 наведено концептуальну діаграму, що зображає досліджувану функцію системи, а саме «Облік записів на прийом до ветеринарного закладу». Метою створення даної діаграми є розробка та візуальне пояснення вимог до створюваної інформаційної технології електронного документообігу. При розробці діаграми було дотримано підхід погляду з боку адміністратора ветеринарного закладу. Створена модель була сконструйована за типом «як буде».

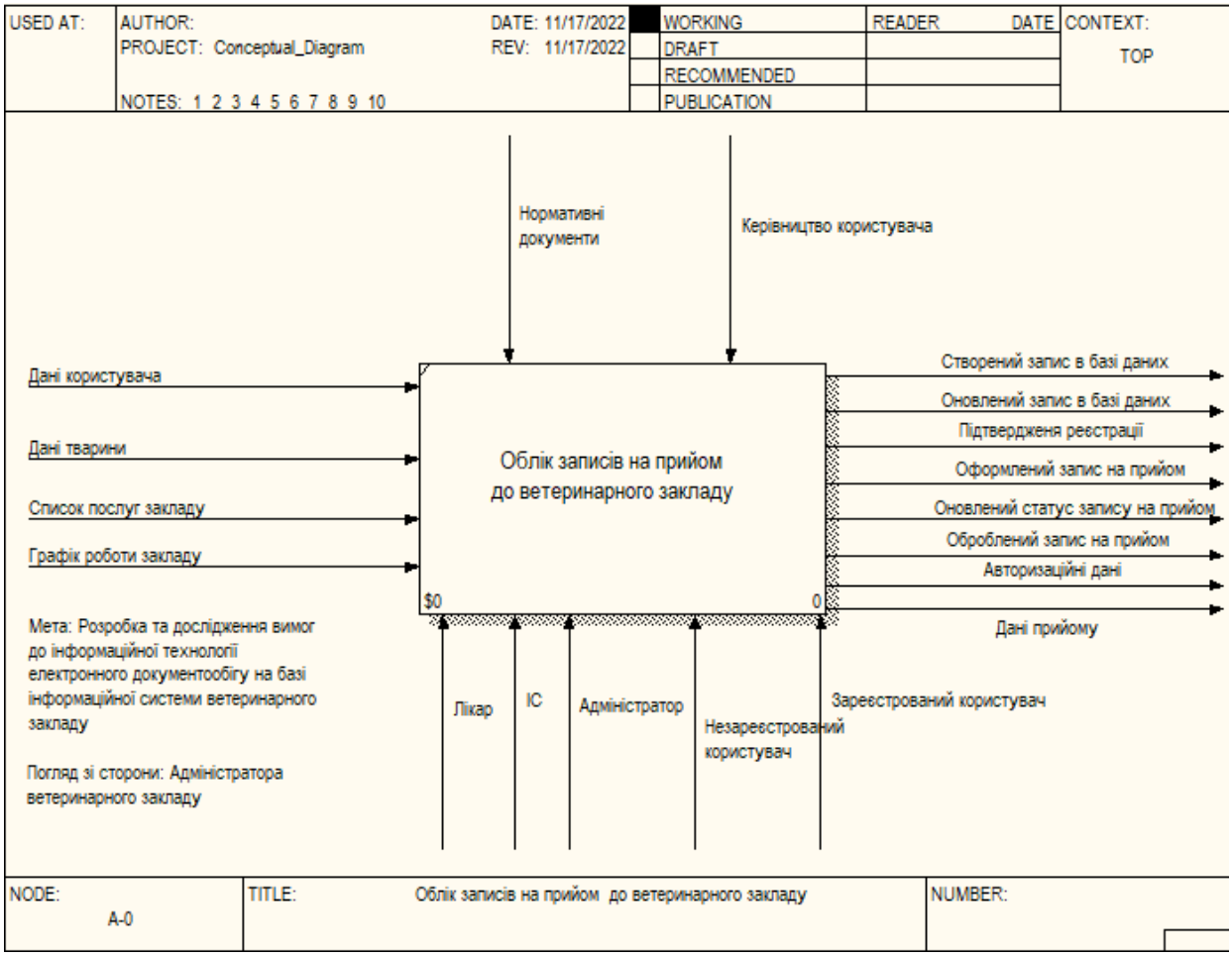

Рисунок 3.4 – Концептуальна діаграма інформаційної технології

Вхідними даними на концептуальній діаграмі представлені дані, які надходять до системи для ведення обліку записів на прийом. Серед цих даних можна виділити дані користувача, які він вводить при створенні облікового запису, дані тварини, які користувач вводить при заповненні форми оформлення запису на прийом. І список послуг клініки та графік роботи закладу, дані яких надає адміністратор закладу при заповненні БД.

Вихідними даними є дані, які отримуються на виході роботи системи. Наприклад, створений запис у базі даних, який створюється після оформлення запису на прийом. Даний запис містить інформацію, яка була введена в поля форми оформлення запису.

Механізмами роботи з системою виступають головні користувачі системи, серед яких можна виділити власне інформаційну систему, адміністратора ветеринарного закладу, користувача, який може буде як зареєстрований, так і незареєстрований та лікар, який надає дані після прийому.

Декомпозиція концептуальної діаграми зображена на рис.3.5.

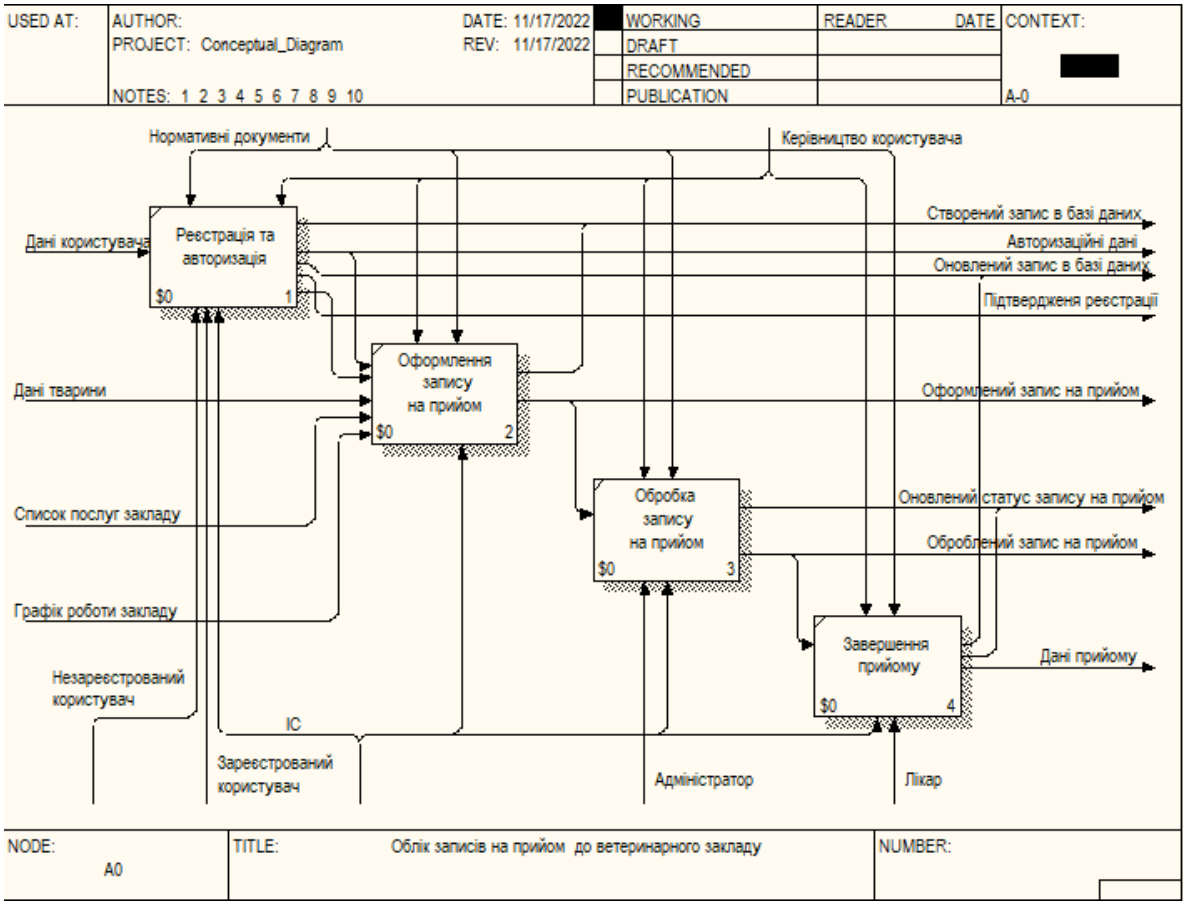

Рисунок 3.5 – Декомпозиція концептуальної діаграми

Головна функція системи була поділена на 4 під-функції:

- «Реєстрація та авторизація». Відповідає за початковий процес реєстрації користувача, тобто введена особових даних, для наступного процесу авторизації, тобто введення даних, які допоможуть увійти до облікового запису. До таких даних відносяться пошта та пароль, за допомогою яких користувача буде ідентифіковано;

- «Оформлення запису на прийом». Функція, що відповідає за процес збереження даних, що були введені у форму оформлення запису на прийом. До таких даних входять наступні – кличка тварини, її вік, порода, тип послуги для прийому, дата прийому і фото тварини за бажанням користувача;
- «Обробка запису на прийом». Функція, за яку відповідає адміністратор системи, тобто процес обробки оформлених записів користувачів, з можливістю майбутньої зміни умов прийому, за бажанням користувача. У випадку, якщо клієнт відмовився від зв'язку з адміністратором система автоматично надає його оформленому запису статус «Оброблений»;
- «Завершення прийому». Функція, яку виконує лікар. Що проводить прийом. Тобто, відкриття особової картки тварини, з наступним записом даних, які були виявлені у процесі прийому. У кінці прийому лікар повинен змінити статус запису на «Завершений» та за необхідності оформити з клієнтом новий запис на прийом на необхідну дату.

Даний порядок під-функцій можна описати наступним чином – спочатку проходить реєстрація та авторизація користувача. У деяких випадках необхідна функція зміни паролю, яка теж присутня у системі. Наступним кроком є оформлення запису на прийом зареєстрованим користувачем, що є процесом вводу даних у поля форми запису на прийом, після чого відбувається процес обробки запису на прийом адміністратором закладу, чи у автоматичному форматі, без його участі. Кінцевим кроком є проведення власне прийому, з введенням даних, які було отримано під час проведення прийому.

4.4. Розробка діаграми варіантів використання інформаційної технології електронного документообігу ветеринарного закладу

Для розробки діаграми варіантів використання було створено «юзкейс» діаграму, тобто діаграму сценаріїв використання системи, що є описом взаємодії учасників, що користуються даною системою. На рисунку 3.6 представлено діаграму використання, що зображує сценарії використання між базовими типами користувачів системи.

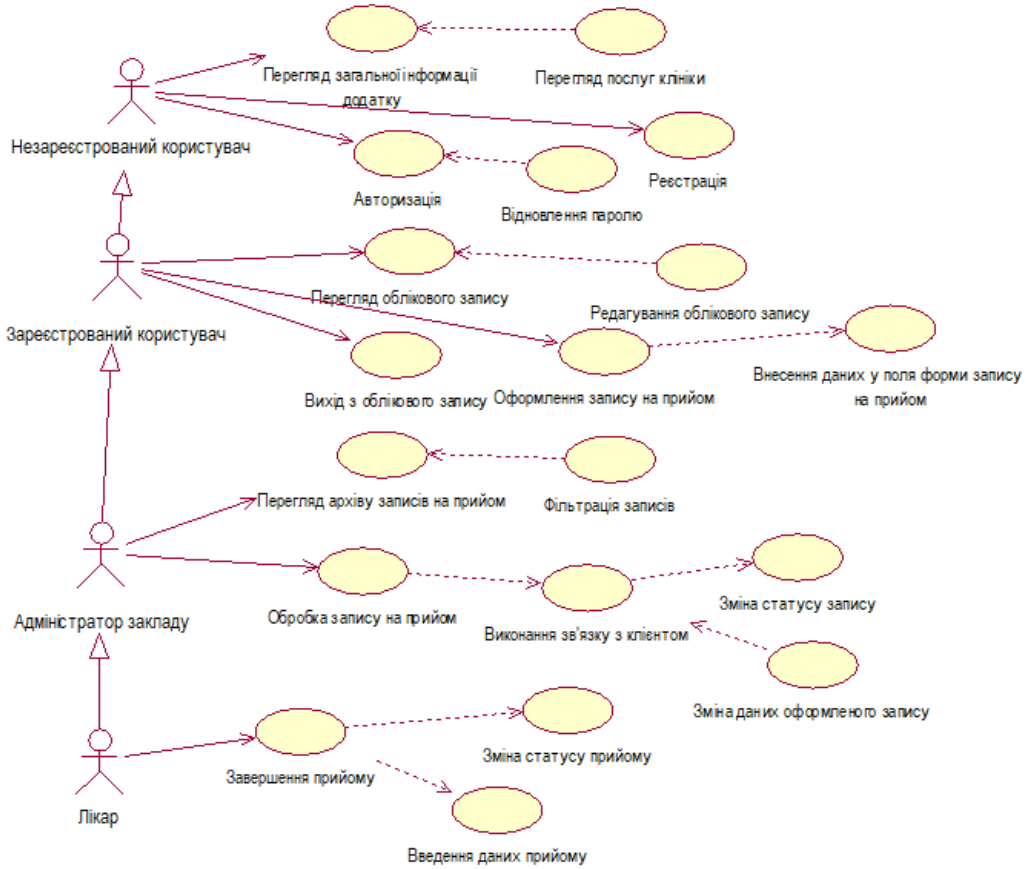

Рисунок 3.6 – Діаграма варіантів використання системи

На діаграмі проглядається збіг акторів з механізмами, що були описані на функціональній моделі, не враховуючи інформаційну систему.

На діаграмі використовуються наступні типи акторів:

1. Незареєстрований користувач. Користувач системи, що не має облікового запису, тобто користувач який тільки почав користуватися веб-додатком. Він має можливість переглянути загальну інформацію, що доступна в додатку, зареєструватися та авторизуватися, при необхідності є можливість змінити пароль;
2. Зареєстрований користувач. Користувач системи, що має особистий обліковий запис, тобто такий, що успішно пройшов процес реєстрації. Для нього доступній обліковий запис користувача з можливістю редагування даних. Він має можливість оформлювати запис на прийом, тобто вводити дані у поля форми для оформлення запису на прийом. Також, для нього доступній вихід з облікового запису;

3. Адміністратор закладу. Особа, яка має спеціальний адміністративний обліковий запис, тобто запис який надає додатковий функціонал користувачеві. Має можливість переглядати архів записів на прийом та сортувати/фільтрувати їх за певним атрибутом. Однією з основних задач є процес обробки запису на прийом, на виході якого повинен оновитися статус запису на прийом. За необхідності адміністратор може змінити умови запису на прийом;
4. Лікар. Особа, яка власне проводить прийоми. Має функціонал адміністратора, тобто для нього доступен архів записів. Він має провести прийом, та завершити запис, тобто змінити його статус та внести дані, які були отримані під час проведення прийому.

На діаграмі представлені наступні прецеденти:

- Перегляд загальної інформації додатку. Можна представити як можливість незареєстрованим користувачем переглядати базову інформацію веб-додатку, що представлена інформаційними сторінками, які описують головні аспекти роботи ветеринарного закладу;
- Перегляд послуг клініки. Можливість користувача додатку переглядати базову інформацію про послуги, які надаються у ветеринарному закладі;
- Реєстрація. Функціонал, який надає доступ незареєстрованому користувачу до можливості реєстрації, тобто введення особистих даних користувача з їх наступним збереженням до бази даних;
- Авторизація. Функціонал зіставлення введених даних для входу з даними уведеними під час процесу реєстрації, для наступного доступу до функцій веб-додатку;
- Відновлення паролю. Редагування паролю, введеного у процесі реєстрації;
- Перегляд облікового запису. Можливість зареєстрованого користувача, після успішного процесу авторизації, переглянути особистий кабінет у обліковому записі;
- Редагування облікового запису. Процес редагування користувальницьких даних у особистому кабінеті;

- Оформлення запису на прийом. Послідовність дій, яка представляє вибір пункту «Оформлення запису», з наступним введенням даних прийому до полів форми запису на прийом. Присутнє введення даних, як у текстові поля, так і у поля з випадаючим списком, чи поля типу дата і час;
- Вихід з облікового запису. Можливість авторизованого користувача припинити користування веб-додатком. Для чого необхідно натиснути на кнопку виходу з облікового запису;
- Перегляд архіву записів на прийом. Сторінка доступу до даних записів на прийом, що представлено загальним списком записів на прийом з можливістю сортувати їх за атрибутами, наявними в даних записах;
- Фільтрація записів. Власне функція сортування записів на прийом за полями таблиці;
- Обробка запису на прийом. Функціонал адміністратору закладу, який передбачає отримання даних щодо оформлення запису на прийом;
- Виконання зв’язку з клієнтом. Необхідність адміністратора ветеринарного закладу виконати зв’язок з користувачем закладу, після оформлення останнім запису на прийом. Користувач веб-додатку під час оформлення запису на прийом може відмовитися від зв’язку з адміністратором;
- Зміна статусу запису. Функціонал зміни статусу запису на прийом у результаті виконання зв’язку з клієнтом. Представлено варіантами «Підтверджений» та «Скасований». У разі відмови користувача від зв’язку з адміністратором статус запису на прийом автоматично змінюється на «Підтверджений»;
- Зміна даних оформленого запису на прийом. Можливість адміністратором змінювати дані, які були введені у процесі оформлення запису. Необхідність редагування даних досягається під час проведенням зв’язку з клієнтом;
- Завершення прийому. Сукупність дій, після якої прийом можна вважати завершеним. До неї відносяться процеси зміни статусу запису на прийом, який приймає відмітку «Проведений». Також до

процесу завершення прийому відноситься, який є прецедентом-включенням і відповідає за функціонал введення даних, які були виявлені під час прийому. До завершення прийому слід віднести можливість лікарем встановлювати дату наступного прийому, тобто виконувати оформлення наступного запису на прийом.

4.5. Розробка моделі потоків даних інформаційної технології електронного документообігу ветеринарного закладу

Контекстна діаграма моделі потоків даних зображена на рис.3.7.

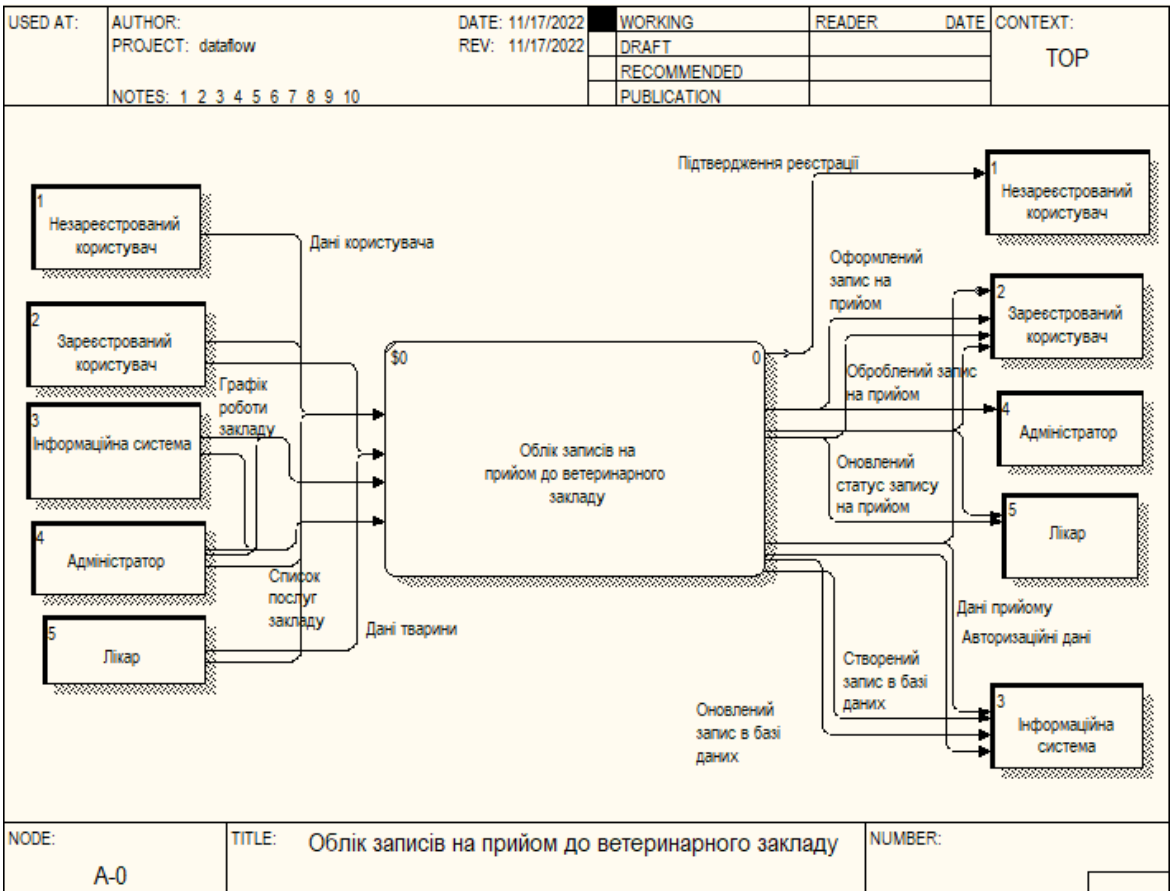

Рисунок 3.7 – Контекстна діаграма моделі потоків даних

На діаграмі представлено 5 типів об’єктів з якими система взаємодіє під час виконання функції, серед яких наступні;

- «Незареєстрований користувач». Користувач, який не має доступу до всіх функцій додатку;

- «Зареєстрований користувач». Користувач, що пройшов процес реєстрації та може бути успішно авторизований у веб-додатку для оформлення запису;
- «Інформаційна система». Власне інформаційна система, яка виконує певні дії з даними під час роботи веб-додатку;
- «Адміністратор». Особа, яка має адміністративний доступ до даних, тобто має доступ до адміністративної панелі з особливим можливостями;
- «Лікар». Особа, що проводить прийоми, та виконує запис даних, які було отримано за час прийому. Змінює остаточний статус прийому на «Закінчений».

4 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ ПРИ РОЗРОБЦІ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

4.1 Логічне й фізичне моделювання даних розроблюваної технології

Для розробки та дослідження інформаційної технології електронного документообігу ветеринарного закладу першочергово слід продумати структуру бази даних системи, яка власне й буде зберігати дані роботи ветеринарного закладу. Для чого й було проведено логічне моделювання розроблюваної бази даних інформаційної технології електронного документообігу ветеринарного закладу.

Результатом проведення логічного моделювання є логічна структура бази даних системи, яка наведена на рис.4.1.

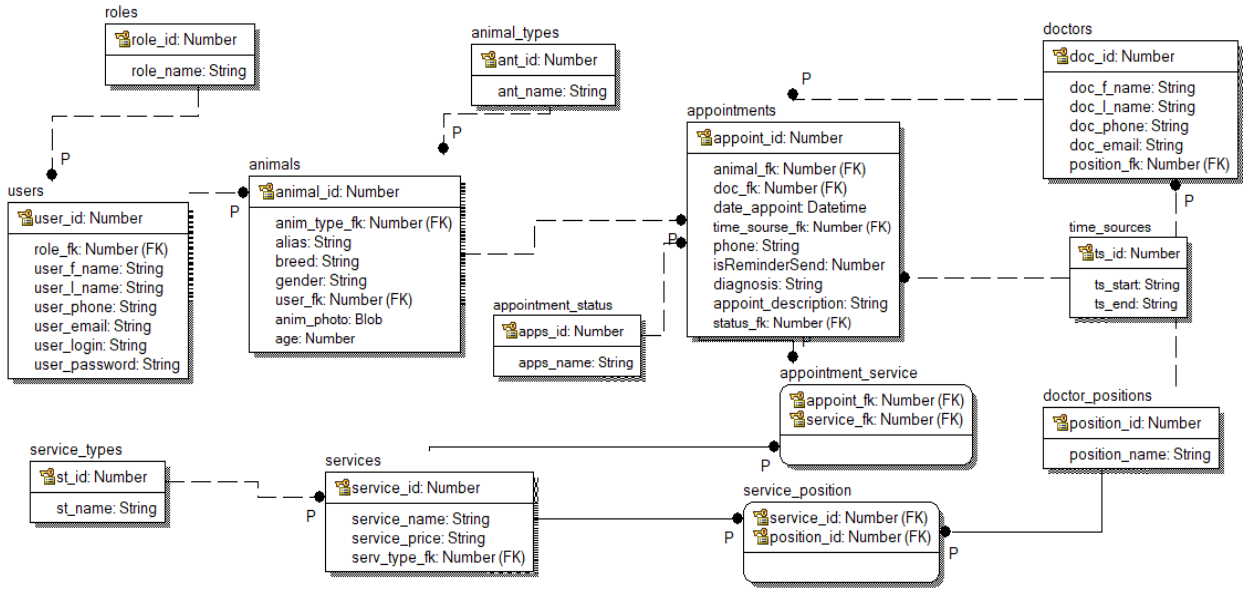

Рисунок 4.1 – Логічна структура бази даних

Кожна з сутностей бази даних відповідає за певний аспект роботи електронного документообігу та системи в цілому.

Сутність «users» відповідає за збереження даних користувача, та потрібна для початкового процесу реєстрації та наступної авторизації. Дані з цієї сутності приймають роль у процесі оформлення запису на прийом та процесі оновлення користувальницьких даних у особистому кабінеті користувача.

Сутність «roles» є сховищем інформації по типам користувачів, які наявні в системі.

Сутність «animals» зберігає дані тварин, які проходять прийоми у ветеринарному закладі. Використовується під час процесу оформлення запису на прийом, адже останній передбачає введення даних тварини, яка потребує прийому. Використовується як інформаційна база при виводі інформації про тварину клієнтові закладу. Дана інформація доступна у особистому кабінеті користувача.

Сутність «animal_types» є зберігаючою сутністю, яка містить дані, щодо типів тварин, яких ветеринарна клініка може обслуговувати. Дана сутність використовується у процесі оформлення запису на прийом, адже з неї дані надходять до випадаючого списку форми для формування запису на прийом.

Сутність «appointments» можна назвати головною сутністю системи, адже вона відповідає за процес прийому у повному обсязі, починаючи від дати і часу прийому і закінчуючи даними, введеними після прийому. Сутність використовується як у процесі оформлення запису, адже зберігає введені користувачем дані, так і в процесі власне прийому, який передбачає введення даних, що були виявлені під час прийому.

Сутність «time_sources» зберігає дані часових інтервалів, записи до яких передбачає робота ветеринарного закладу. Дані використовуються при генерації форми для оформлення запису на прийом. Дані до сутності вносяться адміністратором закладу.

Сутність «doctors» є інформаційною сутністю, яка виконує роль контейнеру з даними лікарів. Дані з сутності використовуються при генерації веб-сторінок додатку, та доступні для перегляду незареєстрованим користувачем.

Сутність «doctor_positions» зберігає дані наявних у ветеринарному закладі професій лікарів. Використовується при оформленні запису на прийом, система на основі введених користувачем даних автоматично визначає лікаря для проведення прийому.

Сутність «services» містить інформацію, щодо послуг, які надаються у ветеринарному закладі, та ціну їх виконання. Дана сутність використовується у процесі оформлення запису на прийом, адже надає дані для цільової форми. На основі інформації в даній таблиці генерується чек за прийом.

Сутність «service_types» зберігає дані типів послуг, що наявні в системі, для структуризації даних.

Сутність «appointment_service» виступає сумісником, тобто сутністю «багато-до-багатьох» між «appointments» та «services». Використовується для можливості групування декількох послуг у один прийом.

Сутність «service_position» об'єднує «services» та «doctor_positions», тобто виступає сутністю «багато-до-багатьох». Розмежовує послуги ветеринарного закладу за професіями лікарів закладу. Використовується під час процесу вибору лікаря для прийому.

Кожна з описаних сутностей виконує ту чи іншу роль у роботі системи та електронного документообігу у даній системі. Однак для повного розуміння роботи даної схеми слід розглянути й фізичну модель. Адже вона враховує специфіку бази даних введеної при створенні проекту, у даному випадку проект створювався під майбутню розробку у MySQL. Структура наведена на рисунку 4.2, та містить інформацію щодо базового типового представлення атрибутів сутностей. Певні поля містять можливість порожнього рядка, адже передбачають заповнення не при створенні запису у БД, а після проведення певних дій, як наприклад проведення прийому у ветеринарному закладі і запису даних, що були виявлені під час прийому.

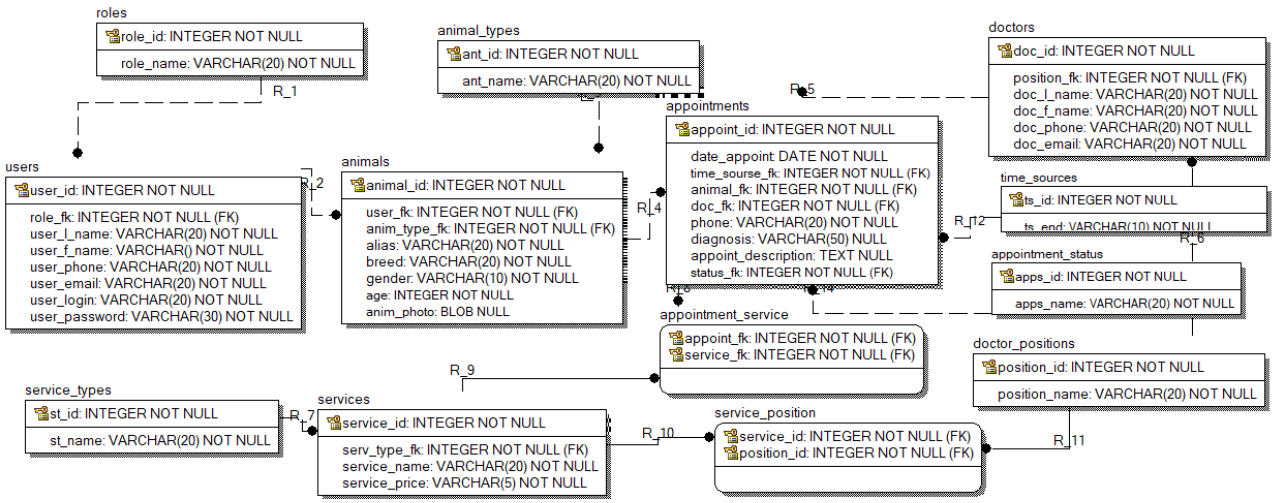

Рисунок 4.2 – Фізична модель структури бази даних

4.2 Генерація бази даних розроблюваної системи

Основним рушієм для взаємодії елементів інформаційної технології електронного документообігу є база даних. У попередньому розділі були описані фізична та логічна її інтерпретації. У цьому розділі показано результат генерації EER-діаграму фізичної моделі бази даних. Результатом генерації є діаграма бази даних, створена у середовищі для роботи з MySQL кодом – Workbench. Діаграма зображена на рис.4.3.

с

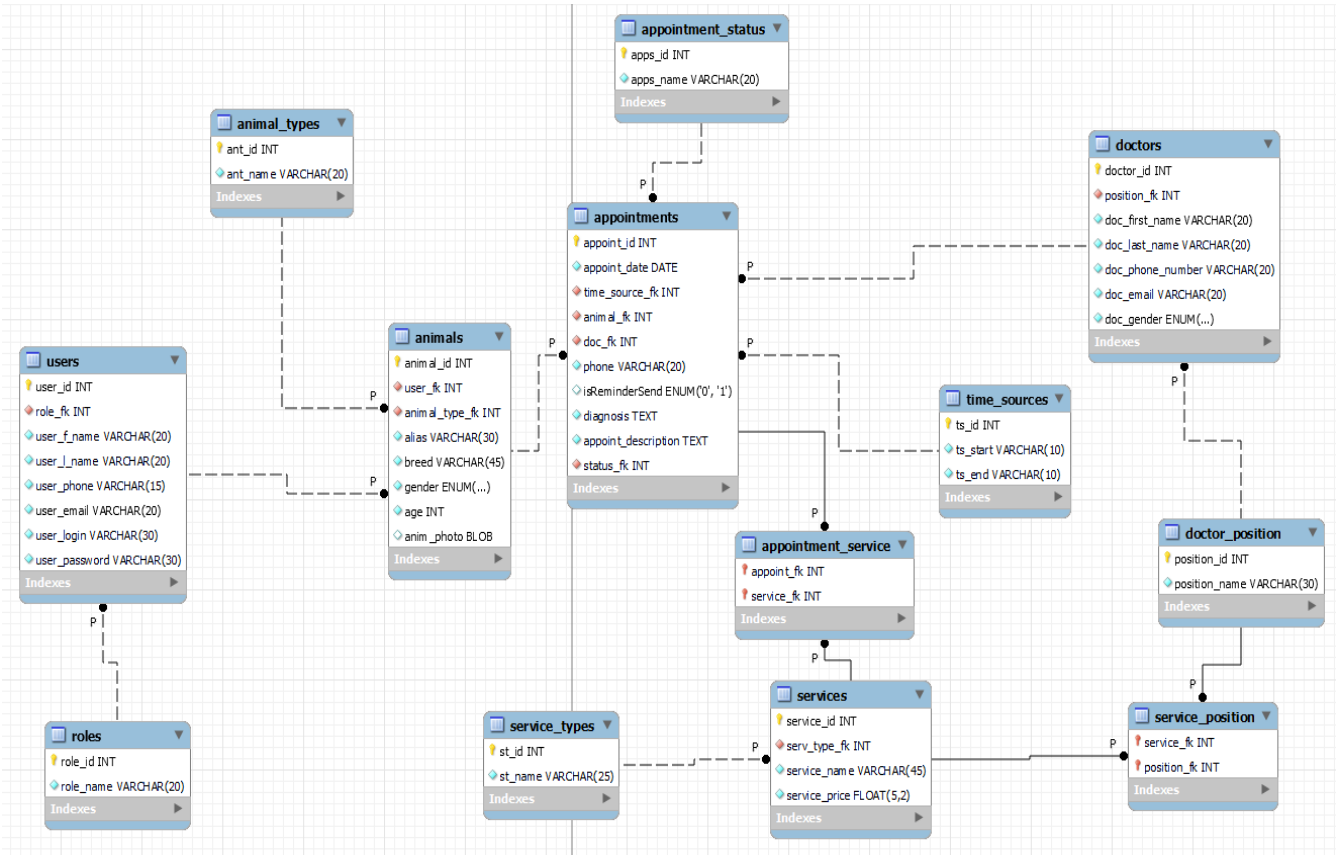

Рисунок 4.3 – EER-діаграма фізичної моделі бази даних

4.3 Розробка алгоритму роботи інформаційної технології на базі веб-додатку

Для розробки алгоритму було взято базову роботу веб-додатку, а саме перехід до попереднього за посиланням і виконання певних дій в залежності від

типу користувача, що зайшов до системи. Базовий алгоритм роботи системи наведено на рис.4.4.

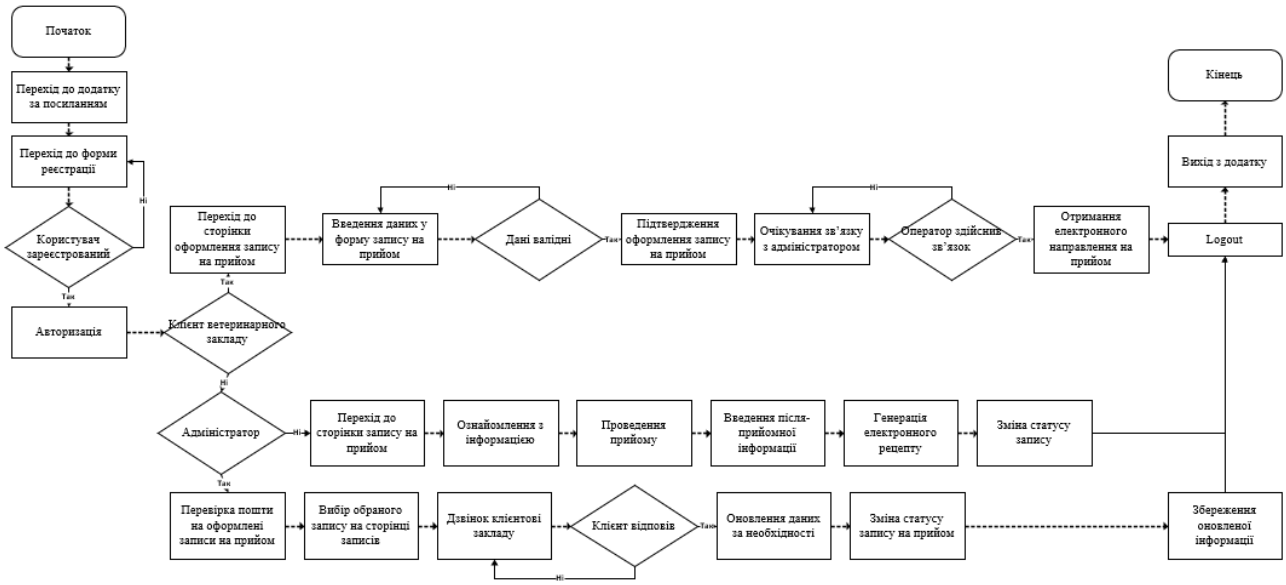

Рисунок 4.4 – Базовий алгоритм роботи системи

Процес роботи алгоритму починається з розпізнавання типу користувача, що доєднався до роботи.

У випадку, коли це звичайний користувач виконується стандартний набір головних дій. Починаючи від процесу переходу до сторінки оформлення запису на прийом, введення даних для оформлення запису, і закінчуючи зв'язком з адміністратором закладу та отриманням електронного направлення на прийом до ветеринарного закладу.

У разі заходу в веб-додаток адміністратора, його зустрічає дещо інший інтерфейс, який необхідний для його роботи з редагуванням та підтвердженням оформлених записів на прийом. Основний алгоритм його дій має наступний формат – отримати на пошту інформацію по оформленому запису на прийом, відкрити дану інформацію у веб-додатку, зв'язатися з користувачем, за необхідності редагувати дані та змінити статус запису на прийом для наступного інформування клієнта системою.

Коли до веб-додатку переходить лікар, то він має провести низку дій для виконання прийому. По-перше відкрити інформацію по прийому, та провести попереднє ознайомлення з електронною карткою тварини, попередніми оглядами, у разі наявності таких. Після проведення прийому лікар повинен

внести до системи інформацію, яка була отримана під час проведення прийому. Інформація зберігається у базі даних для наступного використання у формуванні електронної документації ветеринарного закладу.

Кожен з перерахованих вище типів користувачів має обмежену кількість і тип сторінок, до якого той має доступ. Базова схема доступу до сторінок кожного типу користувачів зображена на рис.4.5.

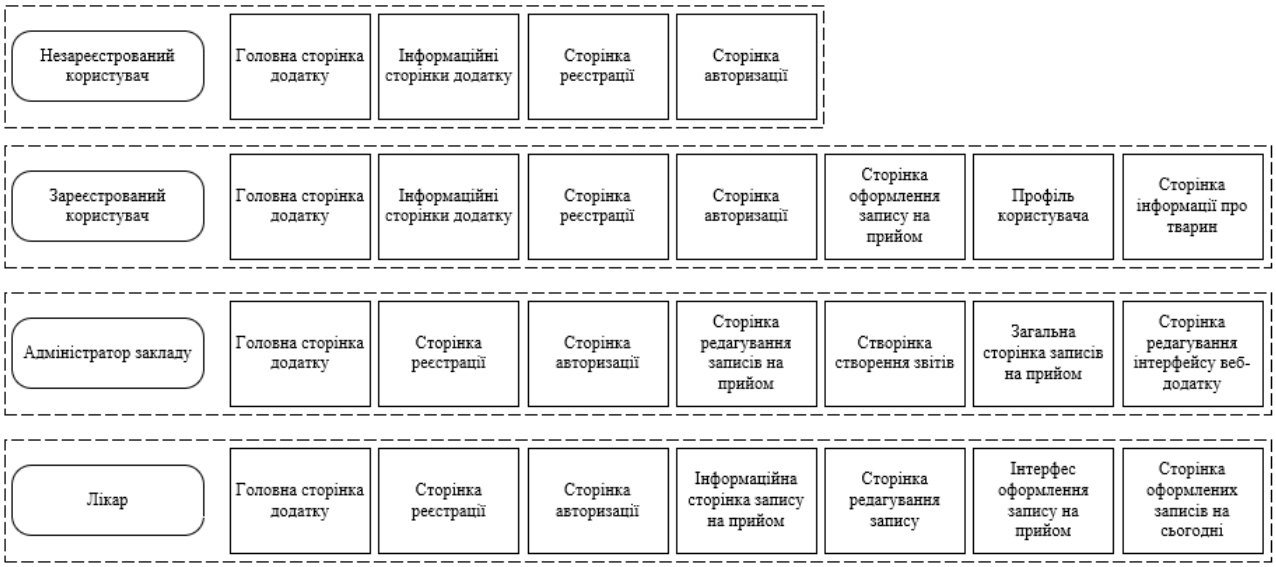

Рисунок 4.5 – Базова схема доступу до сторінок веб-додатку

4.4 Генерація веб-додатку

Для створення веб-додатку використовувалося об’єднання декількох мов програмування. У якості серверної частини веб-додатку використовувався Python з його фреймворком Django. Клієнтською ж частиною виступає JavaScript з його React фреймворком, з використанням стандартних елементів HTML та CSS.

Перше, що необхідно створити у додатках, які управляються Django – це створити ізольоване середовище Python, для того щоб утворити незалежну частину пам’яті, яка може використовувати незалежні бібліотеки та не бути залежною від встановлених на пристрої залежностей між програмними засобами. Після створення такого ізольованого середовища, його необхідно

активувати та продовжувати роботу з додатком з активованим віртуальним середовищем.

Наступним кроком є створення кореневої структури веб-додатку, тобто створити необхідні папки, які будуть відповідати за різні частини системи. На рис.4.6 зображена базова структура папок створюваної технології на базі інформаційної системи.

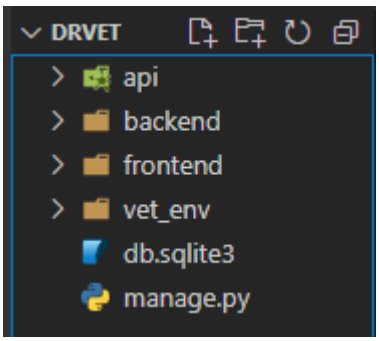

Рисунок 4.6 – Базова структура папок додатку

Як можна побачити, у базовій структурі присутні всі необхідні елементи системи. Починаючи від серверної частини, та закінчуючи об’єднуючою ланкою між сервером та клієнтською стороною додатку.

Для правильної роботи з усіма елементами системи необхідно визначити початкову конфігурацію для URL, тобто базові патерни, за якими система визначає шляхи для переходу між URL. У лістингу 4.1 зображено типовий вигляд початкової конфігурації URL створюваного веб-додатку.

Лістинг 4.1 – Початкова конфігурація між URL

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('api.urls')),
    path('', include('frontend.urls'))
]
```

Даний фрагмент коду визначає базові шляхи для роботи додатку. Тобто, якщо URL містить фрагмент «admin» , необхідно перейти до директиви

«admin.site.urls» та виконати певні дії в залежності від інформації, що йде після фрагменту. Тобто, даний код визначає початкові URL маршрути для маршрутизації веб-додатку.

Наступним процесом виступає механізм створення моделей, тобто програмних реалізацій елементів системи. Для прикладу, у лістингу 4.2 наведено процес створення моделі User, тобто клієнта системи. Дана модель виступає програмним баченням користувача додатку, та необхідна для процесу реєстрації нового клієнта у системі.

Лістинг 4.2 – Створення моделі User

```
class User(models.Model):
    user_f_name = models.CharField(max_length=30, default="")
    user_l_name = models.CharField(max_length=30, default="")
    user_phone = models.CharField(max_length=10, default="")
    user_email = models.CharField(max_length=30, default="")
    user_login = models.CharField(max_length=25, default="")
    user_password = models.CharField(max_length=25, default="")
    created_at = models.DateTimeField(auto_now_add=True)
```

Модель створюється у вигляді класу, та містить всі необхідні поля для наступного зберігання та передачі даних у базу даних. Кожне поле має необхідний тип даних, та базове значення. Останній атрибут є системним, та необхідний для роботи Django, він автоматично відмічає у системі час створення кожного користувача.

Після створення моделі необхідно побудувати механізм для переводу стандартних моделей Django у інші формати. Частіше за все використовуються текстові формати. Даний процес прийнято називати серіалізацією даних. Суттю даного процесу є вибір необхідної моделі наступного переводу у необхідний формат. На лістингу 4.3 наведено фрагмент коду, який показує базовий процес серіалізації моделі користувача з наступним використанням у роботі додатку.

Лістинг 4.3 – Серіалізація моделі користувача

```
from rest_framework import serializers
from .models import User

class UserSerializer(serializers.ModelSerializer):
```

```
    class Meta:
        model = User
        fields = ('id', 'user_f_name', 'user_l_name', 'user_phone',
        'user_email', 'user_login', 'user_password')
class UserView(generics.ListAPIView):
    queryset = User.objects.all()
    serializer_class = UserSerializer
```

В коді можна побачити процес серіалізації, як вибір необхідних атрибутів для перетворення у текстовий формат. У класі «UserView» обрані дані використовуються як база для роботи з ними.

Після необхідного процесу серіалізації, тобто зміни даних необхідно створити клієнтську частину додатку, яка зможе надавати користувачам можливість для вводу необхідних даних, тобто надаватиме клієнтський інтерфейс у вигляд форм для вводу даних. Для створення клієнтської частини використовується бібліотека React. Першим кроком для створення клієнтської частини є створення правильних залежностей між бібліотеками для її роботи. Після чого необхідно створити базову сторінку, яка й надаватиме доступ до React. Базова сторінка посилається на файл «main.js», який відповідає за роботу сервера клієнтської частини.

Базова сторінка React приведена у лістингу 4.4.

Лістинг 4.4 – Базова сторінка React

```
import React, { Component } from "react";
import { render } from "react-dom";
import HomePage from "./HomePage";
export default class App extends React.Component {
    constructor(props) {
        super(props); this.state = {}
    }
    render() {
        return (<div>
            <HomePage />
        </div>)}
}
const appDiv = document.getElementById("app");
render(<App name="andrey" />, appDiv)
```

Як можна побачити з коду, тут містяться посилання на необхідні бібліотеки та домашню сторінку, яка виступає вікном для багатьох сторінок. В цьому і є принцип роботи React – створити головне вікно і надавати необхідні сторінки без їх перезавантаження, просто замінюючи вміст створеного вікна. Це вважається головним плюсом React фреймворку. Код домашньої сторінки, що надає початкові можливості по розмежуванню вікон клієнтської частини наведено у лістингу 4.5.

Лістинг 4.5 – Код домашньої сторінки

```
export default class HomePage extends React.Component {
    constructor(props) {
        super(props)
    }
    render() {
        return (<Router>
            <Routes>
                <Route         path="/"         element={"<p>Hello,
test!</p>"}></Route>
                <Route path="/signin" element={<SignInPage />} />
                <Route path="/login" element={<LoginPage />} />
            </Routes>
        </Router>)
    }}
```

Кожен із зображених «Route» є посиланням на наступний компонент системи за умови, що в URL з’явиться шлях указаний в атрибуті «path». Компоненти є головними елементами екосистеми React, адже на них базується його роботи. Компоненти є по суті HTML-сторінками, що об’єднано з JS-функціоналом. Вони дозволяють використовувати різні бібліотеки для створення будь-якого користувальницького інтерфейсу. А також вони містять можливість зберігати «стан». Під станом розуміється структура коду, яка дозволяє зберігати та оперувати будь-якою інформацією, що надходить до компоненти, нехай це буде інформація ззовні, чи та, що вводиться у поля форми. Даний стан можна використовувати для роботи з функціоналом веб-форм. Є можливість коригувати роботу полів різних типів, починаючи від текстового і закінчуючи полем з міткою. Частина коду стану сторінки реєстрації наведена у лістингу 4.6.

Лістинг 4.6 – Код стану сторінки реєстрації користувача

```
export default class SignInPage extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            user_f_name: "test",
            user_l_name: "test",
            user_phone: "0000000000",
            user_email: "test@gmail.com",
            user_login: "test",
            user_password: "123”, };
```

На фрагменті коду стан зберігає тестову інформацію, яка зберігається за полями форми сторінки. Тобто, в разі надсилання порожнього поля стан додає до його вмісту тестовий текст.

Інтерфейс сторінки створено за допомогою бібліотеки MaterialUI, яка надає можливості по генерації дуже охайних елементів форми за допомогою продуманих елементів та атрибутів. Інтерфейс форми реєстрації наведено на рис.4.7.

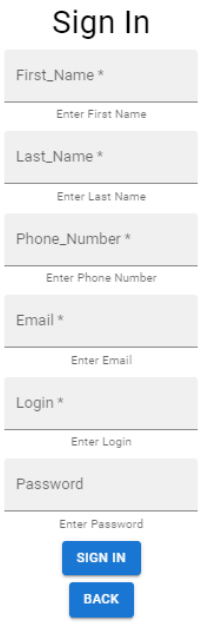

Рисунок 4.7 – Інтерфейс форми реєстрації

4.5 Генерація інформаційної технології електронного документообігу

Головним ідеєю веб-додатку є дослідження та створення інформаційної технології електронного документообігу на основі інформаційної системи ветеринарного закладу.

Електронний документообіг «покликаний» для вирішення проблем стандартного документообігу, тобто паперового. Створення його електронної версії значно прискорить роботу всього ветеринарного закладу. Починаючи від моменту створення користувальницьких карток у електронному форматі і завершуючи наданням електронного направлення на лікування з переліком ліків, які необхідні для лікування.

Основними напрямками використання електронного документообігу у інформаційній системі ветеринарного закладу виступають наступні:

- створення користувальницьких карток, тобто електронного представлення інформації про клієнта;
- введення у систему ветеринарного закладу електронних карток тварин з повною інформаціє про відвідані прийоми, та післяприйомною інформацією;
- генерація електронних направлень на прийом, які надходять користувачеві після підтвердження запису на прийом;
- ведення електронної звітності роботи закладу.

Створення користувальницьких карток відбувається безпосередньо під час процесу реєстрації в системі. Інформація, яку користувач вводить у поля форми для реєстрації зберігається у базі даних з можливістю наступного вибору будь-якого користувача і внесення інформації у підготовлені шаблонні сторінки, для виводу у певному форматі. Приклад шаблону для виведення інформації користувача наведено у лістингу 4.7.

Лістинг 4.7 – Шаблон для виводу інформації про користувача

```
{%  block content  %}
  <p>User</p>
    <ul>
        {%  for u in data  %}
            <li>
```

```
            <p> u.user_f_name </p>
            <p> u.user_l_name </p>
             </li>
        {%  empty  %}
            <p>No user have ben added yet.</p>
        {%  endfor  %}
    </ul>
{%  endblock content  %}
```

Структура є базовою, та потребує доробки, однак її функціонал допомагає вивести на сторінку інформацію про ім’я та прізвище користувача, з наступною можливістю для збереження сторінки у .pdf форматі.

Електронна карта тварини формується з початку оформлення запису на прийом. Дані, що вводяться до форми оформлення запису зберігаються до бази даних. Після збереження даних головна інформація про тварин клієнта надається у його користувальницькому кабінеті. За необхідності інформацію можна завантажити у вигляді електронного документу. Після проведення прийому електронна картка оновлюється, до неї додається інформація про прийом, вказується діагноз, план лікування та після-прийомна інформація.

Електронне направлення є інструментом для інформування користувача про успішне чи неуспішне підтвердження запису на прийом. Після оформлення запису, за необхідності, адміністратор проводить зв’язок з користувачем, уточнює хвилюючі останнього моменти та підтверджує запис на прийом. У цей час на сервері створюється шаблонний документ для інформування користувача при його запис, де вказано час, день, базову інформацію про тварину та успішність підтвердження запису на прийом. Початковий варіант електронного направлення наведено на рис 4.8. Даний документ можна вважати необхідним, адже його відсутність може мати на увазі неможливість лікаря приступити до огляду тварини.

Направление на приём к врачу-ветеринару

Запись №	6
Статус записи	Подтверждённый
Услуга	Первичный клинический осмотр врача офтальмолога
Адрес	Харьков, Харьковская область
Телефон администратора	0661917322
Дата приёма	2021-06-21
Время приёма	09:00-10:30
Врач-ветеринар	Бирюкова Евгения
Специальность врача-ветеринара	Ветеринарный врач офтальмолог
Клиента	Андрей Мояомах
Кличка животного	Сыч
Тип животного	Кот
Порода	Манул
Пол животного	Самец
Цена приёма	150₴

Рисунок 4.8 – Базове представлення електронного направлення

Електронна звітність закладу є важливою частиною роботи ветеринарної клініки. Загальна успішність роботи, виторг закладу, добові звіти – це все, що потребує бізнес для правильної роботи. Електронна звітність базується на вибірках даних з бази даних, з наступним підставленням до шаблонних форм по типу форми представленого у лістингу 4.7.

Функціонал електронного документообігу дозволить з новим поглядом подивитись на ветеринарні заклади, адже від клієнтів тепер не потрібно приходити до закладу для запису на прийом, чи його підтвердження. Реалізація такого функціоналу дозволить охопити значно більше коло можливих клієнтів ветеринарного закладу, за рахунок використання електронного запису.

ВИСНОВКИ

Результатом дослідження та розробки інформаційної технології електронного документообігу ветеринарного закладу є записка до кваліфікаційної роботи. Записка містить основні етапи аналізу та створення інформаційної технології на базі інформаціної системи ветеринарного закладу. Проаналізовано предметну область створюваної системи, для чого було розібрано поняття веб-системи та його зв'язок з інформаційними системами та електронним документообігом. Проаналізовано предметну область, що визначає діяльність компанії для чого було освітлено суть ветеринарної діяльності України та приведено типову схему організаційної структури ветеринарного закладу. Для кожного з працівників клініки було наведено список його задач у інформаційній системі закладу. Проведено аналіз існуючих аналогів системи, та наведено основні їх недоліки, та переваги, які можна використати у розроблюваній системі. Проаналізувавши виявлені проблеми діяльності ветеринарних закладів було наведено кроки для поліпшення їх роботи за рахунок проведення автоматизації системи.

У другому розділі проведено повноцінний огляд методів та технологій, що використовуються в предметній області. У повній мірі розглянуто технологію електронного документообігу та кроки його запровадження у організації. Розглянуто загальні технології реалізації клієнтської частини застосунку, для чого було описано технології HTML, CSS, JavaScript та React, на взаємодії якій і базується робота клієнтської сторони веб-додатку. Наступним кроком були проаналізовані технології реалізації серверної частини додатку, для чого були описані технології Python та Django, зв'язка роботи яких і представлятиме серверну частину системи. Проаналізовано та обрано середовище розробки системи, та надано рекомендації щодо можливої зміни середовища. Розглянуто технології взаємодії клієнта та серверу, для чого було описано роботу протоколу HTTP. У кінці розділу було розглянуто основні засоби управління базами даних, якими користуються на даний момент та обрано типаж бази даних для створення системи.

Третій розділ містить постановку задачі для розробки системи. Постановка задачі містить опис проблем сучасного підходу до функціонування ветеринарного закладу та приблизний опис методів для рішення цих проблем.

При описі рішень було наведено вхідну інформацію, яка надходить до системи у тому чи іншому вигляді.

Наступний розділ містить кроки по розробці інформаційної технології вирішення поставленої задачі. Розроблено системні вимоги для створюваної системи, до яких належить необхідна операційна система, технології клієнтської та серверної частин веб-додатку. Описано архітектуру розроблюваного додатку, алгоритм його функціонування та базову схему електронного документообігу в системі. Розроблено функціональні вимоги до системи за рахунок створення концептуальної діаграми інформаційної технології з використанням стандарту IDEF0. Розроблена діаграма варіантів використання сценаріїв системи та описані основні актори, які взаємодіють з системою.

Розділ з опису прийнятих системних рішень містить інформацію щодо створення інформаційної технології на базі веб-системи ветеринарного закладу. Під цим розуміється процес опису логічного та фізичного моделювання даних розроблюваної системи, результатом якого стали логічна та фізична структури бази даних, готові до їх реалізації. Наступним кроком стала власне реалізація бази даних у обраному середовищі її розробки. Після чого було представлено базовий алгоритм роботи веб-додатку для кожного типу користувачів, та надано базову схему доступу до сторінок кожним з користувачів. Після наведення алгоритму роботи веб додатку та схеми доступу до його сторінок приведені кроки по генерації власне веб-додатку, та шаблони з ідеями генерації частини електронної документації. Наведено міркування з того, як використання електронного документообігу дозволить покращити результати виторгу ветеринарного закладу та збільшити кількість клієнтів, які надійдуть за рахунок перенесення функціоналу електронного запису у онлайн.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Баку А.М. Дослідження інформаційної технології електронного документообігу для впровадження до роботи системи ветеринарного закладу. IX Міжнародна науково-практична конференція “MODERN RESEARCH IN WORLD SCIENCE” у м. Львів, 28-30 листопада, 2022, с. 461-464.

2. Стаття «Електронний документообіг в Україні: як він працює». URL: https://buduysvoe.com/publications/elektronnyy-dokumentoobig-v-ukrayini-yak-vin-pracyuye (дата звернення 03.11.2022).

3. Стаття «Що таке електронний документообіг». URL: https://portfel.ua/shho-take-elektronnij-dokumentoobig/ (дата звернення 03.11.2022).

4. Стаття «У чому відмінність вебдодатків від мобільних додатків». URL: https://brander.ua/blog/u-chomu-vidminnist-vebdodatkiv-vid-mobilnykh-dodatkiv (дата звернення 24.10.2022).

5. Стаття «HTML, CSS і JavaScript у веб». URL: https://vc.ru/flood/50683-html-css-i-javascript-v-vebe-poymut-dazhe-chayniki (дата звернення 24.10.2022).

6. Стаття «Що таке веб додаток? Різниця між сайтом, веб-додатком, SPA і PWA». URL: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/ (дата звернення 24.10.2022).

7. Стаття «Особливості web-додатків». URL: https://sites.znu.edu.ua/webprog/lect/1191.ukr.html (дата звернення 24.10.2022).

8. Стаття «Функція онлайн-запису до лікаря: переваги і недоліки». URL: https://medplatforma.com.ua/article/795-funktsya-on-layn-zapisu-do-lkarya-perevagi-nedolki (дата звернення 18.10.2022).

9. Стаття «Створення структури ветеринарної клініки». URL: https://www.svoydoctor.ru/franchayzing/franshiza/sozdanie-struktury-veterinarnoj-kliniki./ (дата звернення 17.10.2022).

10. Стаття «Організаційна структура ветеринарної медицини та керівництво ветеринарною справою в Україні». URL: https://helpiks.org/3-9073.html (дата звернення 17.10.2022).

11. Огляд CRM-додатку для ветеринарної клініки. URL: https://appointer.ua/programa-dlya-vet-kliniki/ (дата звернення 18.10.2022).

12. Огляд ветеринарних клінік України на порталі PetsToday. URL: https://petstoday.com.ua/clinics/ (дата звернення 18.10.2022).

13. Стаття «Як впливає сервіс онлайн-запису та автоматизації бізнесу на лояльність споживачів?». URL: https://online-zapis.com/servis_onlajn-zapisi_i_avtomatizacii_biznesa (дата звернення 17.10.2022).

14. Стаття «Електронний документообіг в Україні: як він працює». URL: https://buduysvoe.com/ru/publications/elektronnyy-dokumentooborot-v-ukraine-kak-rabotaet (дата звернення 16.11.2022).

15. Стаття «Електронний документообіг в Україні». URL: https://medoc.ua/blog/elektronnij-dokumentoobig-v-ukrani (дата звернення 16.11.2022).

16. Стаття «Електронний документообіг замість паперового». URL: https://buhplatforma.com.ua/article/9413-elektronniy-dokumentooborot-zamst-paperovogo (дата звернення 16.11.2022).

17. Стаття «Що таке електронний документообіг». URL: https://portfel.ua/shho-take-elektronnij-dokumentoobig-r/ (дата звернення 16.11.2022).

18. Стаття «Структура веб-застосунку». URL: http://labaka.ru/likbez/struktura-veb-prilozheniya (дата звернення 25.10.2022).

19. Стаття «Як працює ВЕБ. Клієнт-серверна модель і архітектура веб-застосунків»: https://wiki.merionet.ru/servernye-resheniya/86/kak-rabotaet-web-klient-servernaya-model-i-arhitektura-veb-prilozheniya/ (дата звернення 25.10.2022).

20. Стаття «Як працюють веб-застосунки?». URL: https://habr.com/ru/post/450282/ (дата звернення 26.10.2022).

21. Стаття «Розробка веб-застосунків та чосу вигідно розвивати бізнес у інтернеті.». URL: https://avada-media.ua/services/web-systems/ (дата звернення 25.10.2022).

22. Стаття «Початок роботи з Інтернетом. Основи HTML». URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics (дата звернення 17.10.2022).

23. Стаття «Мова HTML: що це таке і як він працює». URL: https://skillbox.ru/media/code/chto_takoe_html/ (дата звернення 17.10.2022).

24. Стаття «Початок роботи з Інтернетом. Основи CSS». URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/CSS_basics (дата звернення 17.10.2022).

25. Стаття «SEO Wiki - CSS». URL: https://wiki.rookee.ru/css/ (дата звернення 17.10.2022).

26. Стаття «Що таке JavaScript простими словами». URL: https://questions.students-library.com/library/lecture/read/91381-cto-takoe-javascript-prostymi-slovami (дата звернення 17.10.2022).

27. Стаття «Початок роботи з Інтернетом. Основи JavaScript». URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/JavaScript_basics (дата звернення 17.10.2022).

28. Стаття «Основи React: все, що потрібно знати для початку роботи». URL: https://habr.com/ru/company/ruvds/blog/343022/ (дата звернення 17.10.2022).

29. Стаття «React.js для новачків у програмуванні: що це як влаштований і навіщо потрібен». URL: https://skillbox.ru/media/code/reactjs-dlya-novichkov-v-programmirovanii-chto-eto-kak-ustroen-i-zachem-nuzhen/ (дата звернення 17.10.2022).

30. Стаття «Розуміємо React за 5 хвилин». URL: https://medium.freecodecamp.org/learn-react-js-in-5-minutes-526472d292f4 (дата звернення 17.10.2022).

31. Стаття «Перші кроки у програмуванні веб-сайтів на стороні сервера». URL: https://developer.mozilla.org/ru/docs/Learn/Server-side/First_steps/Introduction (дата звернення 17.10.2022).

32. Стаття «Що таке Python простими термінами». URL: https://vc.ru/u/624860-iqbi/347395-chto-takoe-python-prostymi-slovami (дата звернення 17.10.2022).

33. Стаття «Що таке Python?». URL: https://aws.amazon.com/ru/what-is/python/ (дата звернення 17.10.2022).

34. Стаття «Мова Python: що це таке, як користуватися і де використовувати». URL: https://legoteacher.ru/programmirovanie-na-python/yazyk-python/ (дата звернення 17.10.2022).

35. Метиз Е. Вивчаємо Python: програмування ігор, візуалізація даних, веб-додатки. 3-е видання : навч.посіб. Санкт-Петербург : Пітер, 2022. 512 с.

36. Стаття «Серверне програмування веб-сайтів. Django початок». URL: https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/Introduction (дата звернення 24.10.2022).

37. Стаття «Ефективний Django. Частина 1». URL: https://habr.com/ru/post/240463/ (дата звернення 17.10.2022).

38. Стаття «Вибираємо найзручніший редактор коду Python». URL: https://habr.com/ru/company/skillfactory/blog/521838/ (дата звернення 17.10.2022).

39. Стаття «Веб-технології для розробників. Огляд протоколу HTTP.». URL: https://developer.mozilla.org/ru/docs/Web/HTTP/Overview (дата звернення 25.10.2022).

40. Стаття «Що таке база даних?». URL: https://www.oracle.com/cis/database/what-is-database/ (дата звернення 25.10.2022).

41. Стаття «Що таке СУБД – функції і класифікація?». URL: https://ktonanovenkogo.ru/voprosy-i-otvety/subd-chto-ehto-takoe.html (дата звернення 26.10.2022)