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)_____

Дослідження методів та засобів розробки інформаційної технології документообігу (на прикладі ветеринарної клініки)
(тема)

Виконала:

студентка _2_ курсу, групи ____ІТПм-22-2_____

_____Соляник А.Р._____
(прізвище, ініціали)

Спеціальність 122 – Комп'ютерні науки_____
(код і повна назва спеціальності)

Тип програми _____освітньо-професійна_____

Освітня програма __Інформаційні технології____
____проектування_____
(повна назва освітньої програми)

Керівник _____проф. Калита Н.І._____
(посада, прізвище, ініціали)

Допускається до захисту

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

2024 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.

13.01.2024 _____ *Соляник А.Р.*
(дата, підпис, прізвище студента/-ки)

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.
Керівник кваліфікаційної роботи _____ *проф. Калита Н.І.*

Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.
Керівник кваліфікаційної роботи _____ *проф. Калита Н.І.*

Попередній захист проведено 15 січня 2024 р.
Керівник кваліфікаційної роботи _____ *проф. Калита Н.І.*

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 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _____Соляник Анастасії Романівні_____
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів та засобів розробки інформаційної технології документообігу (на прикладі ветеринарної клініки)_____
затверджена наказом університету від 20.11.2023 р. № _1373Ст_
2. Термін подання студентом роботи до екзаменаційної комісії 17 січня 2024 р.
3. Вихідні дані до роботи _Дослідити та розробити інформаційну технологію документообігу для ветеринарної клініки. Перелік використовуваних програмних засобів: ОС Microsoft Windows 10, Visual Studio 2022, Sql Server Management Studio 2019, WebStorm 2023.3.2, Android Studio. Технічне забезпечення: IBM-сумісний ПК з ЦП Intel Core i3/AMD аналог та вище._____
4. Перелік питань, що потрібно опрацювати в роботі _4.1 Вступ, 4.2 Аналіз предметної області, 4.2.1 Опис сучасного стану розвитку інформаційних систем та технології документообігу, 4.2.1.1 Опис інформаційних систем, 4.2.1.2 Опис технології документообігу, 4.2.2 Аналіз застосування досліджуваної технології документообігу в існуючих системах, 4.2.2.1 DocuWare, 4.2.2.2 M-Files, 4.2.2.3 M.E.Doc, 4.2.2.4 Висновки, 4.3 Постановка задачі на дослідження, 4.4 Моделювання процесів обробки інформації, 4.4.1 Розробка функціональних вимог до інформаційної технології документообігу, 4.4.2 Розробка моделі потоків даних інформаційної технології документообігу, 4.4.3 Логічне моделювання даних, 4.5 Дослідження методів вирішення задачі, 4.5.1 Математичний опис методів рішення задачі документообігу, 4.5.1.1 Загальний опис, 4.5.1.2 Математичний опис задачі генерації файлів, 4.5.2 Аналіз методів генерації файлів, 4.5.2.1 Методи генерації файлів у форматі pdf, 4.5.2.2 Методи генерації файлів у форматі docx, 4.5.3 Аналіз методів збереження цілісності файлів, 4.6 Огляд методів та технологій проєктування та розробки застосунку, 4.6.1 Технології реалізації серверної частини застосунку, 4.6.1.1 Огляд мови програмування C#, 4.6.1.2 Огляд технології ASP Web API, 4.6.2 Технології доступу до даних, 4.6.3 Технології реалізації клієнтської частини застосунку, 4.6.3.1 Огляд технології HTML, 4.6.3.2 Огляд технології CSS, 4.6.3.3 Огляд мови програмування JavaScript, 4.6.3.3 Огляд технології

React, 4.6.4 Технології реалізації мобільного застосунку, 4.6.4.1 Огляд мови програмування Kotlin, 4.6.4.2 Огляд технології Retrofit, 4.6.5 Опис архітектури програмної системи, 4.6.6 Розробка системних вимог до інформаційної технології документообігу, 4.7 Опис прийнятих проєктних рішень, 4.7.1 Генерація бази даних, 4.7.2 Генерація інформаційної технології, 4.8 Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій 5.1 Контекстна діаграма системи документообігу ветеринарної клініки (1 аркуш формату А4), 5.2 Діаграма декомпозиції, яка представляє функції системи (1 аркуш формату А4), 5.3 Діаграма потоків даних функції «Зареєструватися та авторизуватися» (1 аркуш формату А4), 5.4 Діаграма потоків даних функції «Записати на прийом» (1 аркуш формату А4), 5.5 Діаграма потоків даних функції «Вести електронну медкартку» (1 аркуш формату А4), 5.6 Діаграма потоків даних функції «Виписати направлення» (1 аркуш формату А4), 5.6 Діаграма потоків даних функції «Супроводжувати дослідження» (1 аркуш формату А4), 5.8 Діаграма потоків даних функції «Замовлення медикаментів» (1 аркуш формату А4), 5.9 Логічна структура бази даних (1 аркуш формату А4).

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання кваліфікаційної роботи	20.11.23	виконано
2	Проведення аналізу предметної області системи	21.11.23 - 26.11.23	виконано
3	Постановка задачі на розробку та дослідження	27.11.23 - 30.11.23	виконано
4	Опис методів рішення задачі документообігу	01.12.23 - 07.12.23	виконано
5	Вибір технологій для реалізації вебзастосунку	08.12.23 - 09.12.23	виконано
6	Опис прийнятих проектних рішень при розробці технології	10.12.23 - 14.12.23	виконано
7	Розробка алгоритму роботи інформаційної технології	15.12.23 - 30.12.23	виконано
8	Оформлення пояснювальної записки	31.12.23 - 13.01.24	виконано
9	Представлення на рецензування	13.01.24	виконано
10	Представлення роботи до ЕК	15.01.24	виконано

Дата видачі завдання _20_ __листопада__2023 р.

Студент _____ _____Соляник А.Р._____
(підпис)

Керівник роботи _____ _____проф. Калита Н.І._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 85 стор., 25 рис., 1 табл., 2 дод., 34 джерела.

ASP WEB API, C#, JAVASCRIPT, KOTLIN, MS SQL SERVER, REACT, ВЕБЗАСТОСУНОК, ВЕТЕРИНАРНА КЛІНІКА, ДОКУМЕНТООБІГ, ДОСЛІДЖЕННЯ, ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДОКУМЕНТООБІГУ.

Об’єкт дослідження – інформаційна технологія документообігу для ветеринарних клінік.

Предмет дослідження – методи та засоби розробки інформаційної технології документообігу, зокрема на прикладі ветеринарної клініки. Дослідження спрямоване на аналіз та оцінку різних підходів до створення систем електронного документообігу, які дозволяють ефективно та безпечно управляти документами в контексті ветеринарної медицини.

Мета роботи – дослідити методи та засоби розробки інформаційної технології документообігу на прикладі ветклініки, оцінка та визначення можливостей впровадження технології електронного документообігу в контексті ветеринарної клініки.

Методи дослідження – системний підхід, методи структурного аналізу та моделювання, проєктування архітектури програмного забезпечення, проєктування структури зберігання даних, UI дизайн системи, методи створення функціональних застосунків.

У роботі проведено аналіз предметної області, де описано сучасний стан розвитку інформаційних систем та технології документообігу. Для визначення основних кроків та задач автоматизації проводиться постановка задачі. Дослідження методів вирішення задачі проводиться з метою вивчення та оцінки різних підходів та стратегій, які можуть бути застосовані для вирішення проблеми електронного документообігу.

Область застосування: ветеринарні клініки.

ABSTRACT

Master's thesis: 85 pages, 25 figures, 1 tables, 2 appendices, 34 titles.

ASP WEB API, C#, JAVASCRIPT, KOTLIN, MS SQL SERVER, REACT, WEB APPLICATION, VETERINARY CLINIC, DOCUMENT FLOW, RESEARCH, DOCUMENT FLOW INFORMATION TECHNOLOGY.

The object of the research is information technology for document management in veterinary clinics.

The subject of the research is the methods and tools for developing information technology for document management, particularly with the example of a veterinary clinic. The research aims to analyze and evaluate various approaches to creating electronic document management systems that enable efficient and secure document handling in the context of veterinary medicine.

The purpose of the work is to investigate the methods and means of developing the information technology of document management on the example of a veterinary clinic, to assess and determine the possibilities of implementing electronic document management technology in the context of a veterinary clinic.

Research methods – system approach, methods of structural analysis and modeling, software architecture design, data storage structure design, system UI design, methods of creating functional applications.

The work analyzes the subject area, which describes the current state of development of information systems and document management technology. To determine the main steps and tasks of automation, a problem statement is made. The study of problem solving methods is conducted with the aim of studying and evaluating various approaches and strategies that can be used to solve the problem of electronic document management.

Field of application: veterinary clinics.

ЗМІСТ

ВСТУП

У сучасному інформаційному суспільстві, де технологічні інновації надають нові можливості для оптимізації різних областей, важливим аспектом є удосконалення систем управління документами. Однією з важливих областей, яка вимагає вдосконалення у цьому контексті, є ветеринарна медицина.

Актуальність даного дослідження обумовлена нагальною потребою в удосконаленні систем управління документами у ветеринарних клініках. Ветеринарні клініки, як і інші області, стикаються з великим обсягом документації, в тому числі медичних записів, рецептів, та інших важливих документів. Зараз у ветеринарних клініках використовують або паперовий документообіг, або системи документообігу, які інтегруються з іншими системами управління ветеринарними клініками. Системи документообігу ветклінік існують, але їх не так багато, і зазвичай вони призначені для працівників клінік, в деяких клієнти можуть лише переглядати медичні картки. Запис на прийом для власників, якщо і є, це просто форма для зв'язку з клінікою, де адміністратор сам потім записує на прийом.

Об’єкт дослідження – інформаційна технологія документообігу для ветеринарних клінік.

Предметом дослідження є методи та засоби розробки інформаційної технології документообігу у вигляді інформаційної системи, зокрема на прикладі ветеринарної клініки.

Методи дослідження – системний підхід, методи структурного аналізу та моделювання, проєктування архітектури програмного забезпечення (ПЗ), проєктування структури зберігання даних, UI дизайн системи, методи створення функціональних застосунків.

Метою кваліфікаційної роботи є дослідити методи та засоби розробки інформаційної технології документообігу на прикладі ветеринарної клініки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис сучасного стану розвитку інформаційних систем та технології документообігу

1.1.1 Опис інформаційних систем

Інформаційна система (ІС) – це система, яка забезпечує збір, зберігання, обробку, аналіз та надання інформації з метою забезпечення ефективності управління та прийняття рішень в певній діяльності чи організації. Іншими словами, це комплекс взаємопов'язаних елементів, що працюють разом для обробки інформації та вирішення конкретних завдань [1].

Основні компоненти інформаційної системи включають:

– люди – користувачі, які взаємодіють з системою, вводять дані, отримують результати обробки інформації;

– дані – вхідні та вихідні дані, які обробляються системою. Дані можуть бути структурованими (наприклад, таблиці в базі даних) або неструктурованими (наприклад, текстові документи);

– процеси – алгоритми та програми, що виконують обробку даних та вирішують конкретні завдання;

– обладнання – апаратні засоби, такі як комп'ютери, сервери, мережеве обладнання, які використовуються для забезпечення функціонування системи;

– споживачі – організації, підприємства або індивідуали, які використовують інформаційну систему для вирішення своїх завдань та потреб;

– вивід – результати обробки інформації, які подаються користувачам у зручній формі (наприклад, звіти, графіки, екрани).

Інформаційні системи можуть бути різних рівнів складності, від простих систем обліку до великих корпоративних інфраструктур, які обслуговують тисячі користувачів. Основна мета інформаційної системи – забезпечення швидкого та

ефективного доступу до інформації для підтримки прийняття рішень і оптимізації бізнес-процесів.

Елементи інформаційної системи виконують різні функції [1]:

– введення (Input) – збір даних або введення інформації в систему. Це може бути введення даних користувачем або автоматичний збір з датчиків чи інших джерел;

– обробка (Processing) – обробка даних за допомогою програм та алгоритмів для отримання корисної інформації. Цей етап включає аналіз, обчислення та інші операції;

– виведення (Output) – представлення результатів обробки у зручній формі для користувача або інших систем. Це може бути виведення на екран, друк, відправлення повідомлень тощо;

– зберігання (Storage) – збереження даних для подальшого використання. Це може бути внутрішнє зберігання на жорстких дисках, зовнішні сховища або хмарні сервіси;

– керування (Control) – керування роботою інших компонентів системи, забезпечення взаємодії між ними та контроль за процесами.

Інформаційні системи використовуються в різних областях, таких як бізнес, наука, охорона здоров'я, освіта та інші, для оптимізації процесів та підвищення ефективності управління. Їх розробка та використання визначаються конкретними потребами та завданнями організацій чи користувачів.

1.1.2 Опис технології документообігу

Технологія документообігу (Document Workflow Technology) використовується для автоматизації процесів обробки та обігу документів у великих організаціях або бізнес-середовищах. Ця технологія дозволяє ефективно керувати створенням, редагуванням, обігом, зберіганням та вивченням документів

в електронній формі, що сприяє поліпшенню робочих процесів та зменшенню витрат часу та ресурсів [2].

Основні характеристики технології документообігу включають:

– електронний обіг документів. Замість традиційного паперового обігу документів, технологія документообігу дозволяє створювати, редагувати та розсилати документи в електронній формі;

– автоматизація робочих процесів. Системи документообігу дозволяють автоматизувати різноманітні етапи обробки документів, такі як підписання, схвалення, розсилка тощо. Це поліпшує ефективність робочих процесів та дозволяє швидше приймати рішення;

– зберігання та пошук документів. Електронні системи забезпечують безпечне зберігання документів і дають можливість швидкого та зручного пошуку необхідної інформації;

– автоматичне сповіщення та нагадування. Системи можуть автоматично генерувати повідомлення та нагадування для учасників процесу документообігу, що сприяє своєчасній обробці документів;

– безпека даних. Технології документообігу враховують питання безпеки даних, забезпечуючи обмін інформацією захищеним та конфіденційним способом;

– інтеграція з іншими системами. Сучасні системи документообігу можуть легко інтегруватися з іншими бізнес-системами, такими як електронні поштові сервіси, CRM-системи, ERP-системи тощо;

– аналітика та звітність. Технології документообігу надають інструменти для аналізу та звітності про обіг документів, що дозволяє оцінювати ефективність робочих процесів та виявляти можливості для оптимізації.

Технології документообігу використовуються в різних областях, включаючи фінанси, охорону здоров'я, освіту, області громадської служби та інші. Також ця технологія допомагає покращити ефективність роботи з документами, зменшити час на обробку та зберігання інформації, а також полегшує відстеження статусу та контроль над робочим процесом.

Використання технології документообігу в ветеринарних клініках може мати значущий позитивний вплив на ефективність та якість надання ветеринарних послуг. Актуальність використання цієї технології в ветеринарних закладах полягає у наступному:

– електронна історія хвороб. Технологія документообігу дозволяє ветеринарним клінікам переходити від паперових медичних карток до електронних історій хвороб. Це полегшує доступ до інформації про пацієнтів, зменшує ризик помилок та покращує управління медичною інформацією;

– електронні форми та документи. Використання електронних форм дозволяє легко заповнювати, підписувати та обмінюватися різними документами, такими як лікувальні плани, рецепти тощо. Це спрощує адміністративні процеси та підвищує точність документообігу;

– автоматизація процесів. Ветеринарні клініки можуть використовувати технологію документообігу для автоматизації процесів, таких як прийом пацієнтів, замовлення ліків, облік лікування та інші. Це допомагає зменшити час на обробку інформації та уникнути помилок;

– системи управління та планування. Технологія документообігу може включати системи управління прийомами, робочим графіком, складами та іншими аспектами керування клінікою. Це сприяє ефективній організації робочих процесів;

– безпека та конфіденційність даних. Врахування стандартів безпеки даних допомагає забезпечити конфіденційність медичної інформації пацієнтів та зменшити ризики порушень безпеки;

– можливість віддаленого доступу. Технології документообігу дозволяють ветеринарним фахівцям отримувати доступ до необхідної інформації з будь-якого місця, що особливо корисно в сучасних умовах, коли важливо бути мобільним та гнучким;

– легкість спілкування з клієнтами. Сучасні системи можуть включати електронні канали спілкування з клієнтами, такі як електронні записи на прийом,

нагадування про призначення та інші сервіси, що полегшують взаємодію з власниками тварин.

Використання технології документообігу відкриває можливості для ефективнішого управління клінікою, поліпшення якості обслуговування, забезпечення безпеки даних та підвищення рівня задоволення клієнтів.

1.2 Аналіз застосування досліджуваної технології документообігу в існуючих системах

1.2.1 DocuWare

DocuWare – це хмарне програмне забезпечення як постачальник послуг. Програмне забезпечення DocuWare забезпечує управління документами, сховище та функції автоматизації робочих процесів.[3]

Програмне забезпечення для керування документами DocuWare забезпечує інтелектуальний цифровий робочий процес і контроль документів для значного підвищення продуктивності без потреби в IT-ресурсах [3].

На рисунку 1.1 зображено приклад редагування документу за допомогою цього програмного забезпечення.

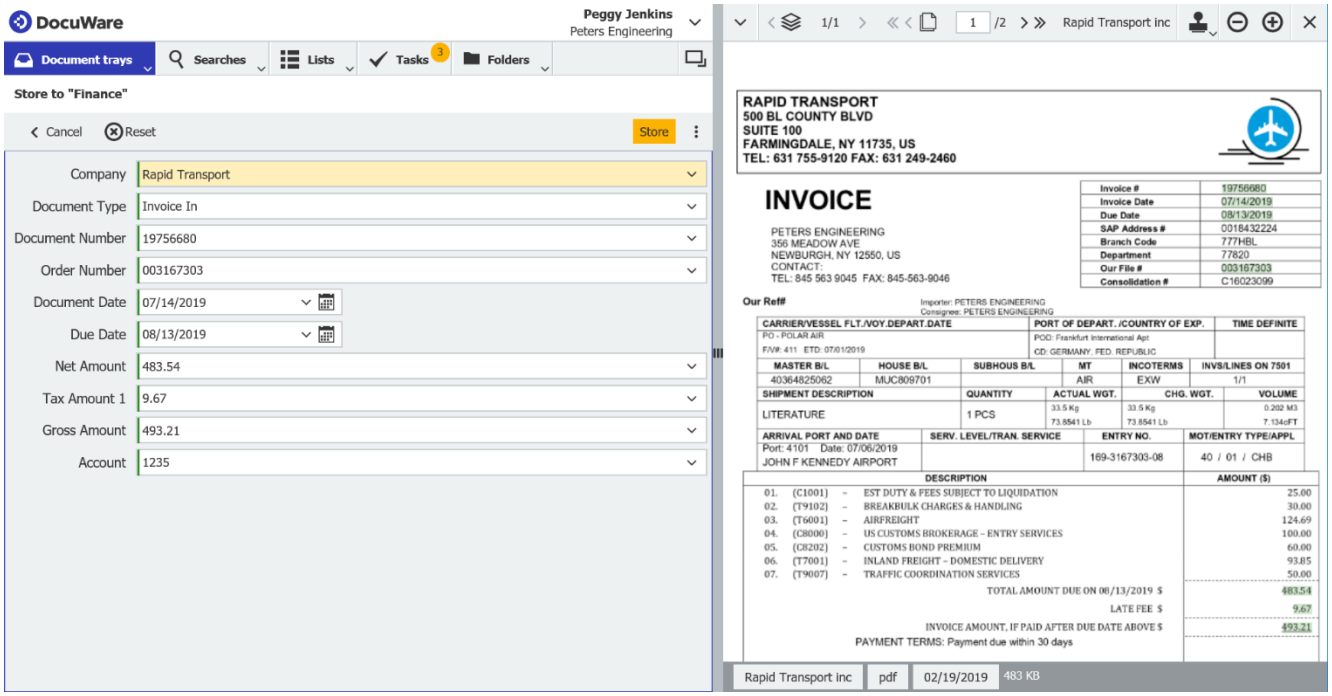


Рисунок 1.1 – Редагування документу у DocuWare

Ключові характеристики DocuWare:

– зберігання документів в електронному вигляді з можливістю створення структурованих архівів;

– автоматизація та оптимізація бізнес-процесів за допомогою потоків роботи та завдань;

– можливість інтеграції з іншими системами, такими як ERP, CRM, електронна пошта та інші;

– заходи безпеки, такі як шифрування та контроль доступу, для забезпечення конфіденційності інформації;

– можливість використання електронного підпису для підтвердження автентичності документів;

– зручний пошук та фільтрація документів за різними критеріями та метаданими;

– можливість отримання доступу до документів та робочих процесів через мобільні пристрої;

– співвідношення з нормативними вимогами та стандартами щодо електронного зберігання даних.

1.2.2 M-Files

M-Files – це система для управління корпоративним контентом, що дозволяє перетворити підхід до управління, захисту та обміну інформацією на підприємстві завдяки організації та обробці контенту на основі його змісту, а не місця зберігання. M-Files може бути розгорнутий на власному устаткуванні, у «хмарі» або в гібридному середовищі для підвищення продуктивності та якості, а також для дотримання норм та стандартів у регульованих областях. M-Files забезпечує повнофункціональний вебдоступ для всіх видів браузерів, а також мобільний доступ для платформ iOS, Android [4].

Інтерфейс користувача M-Files (рис.1.2) складається з чотирьох основних частин [5]:

– область завдань (A) ліворуч містить часто використовувані команди та комбінації клавіш;

– права панель (B) відображає вкладки «Метадані», «Попередній перегляд», «Фільтри» та «Закріплені»;

– панель швидкого пошуку (C) у верхній частині дозволяє шукати документи та інші об’єкти в M-Files;

– область списку (D) у центрі містить список видів, об’єктів і результатів пошуку.

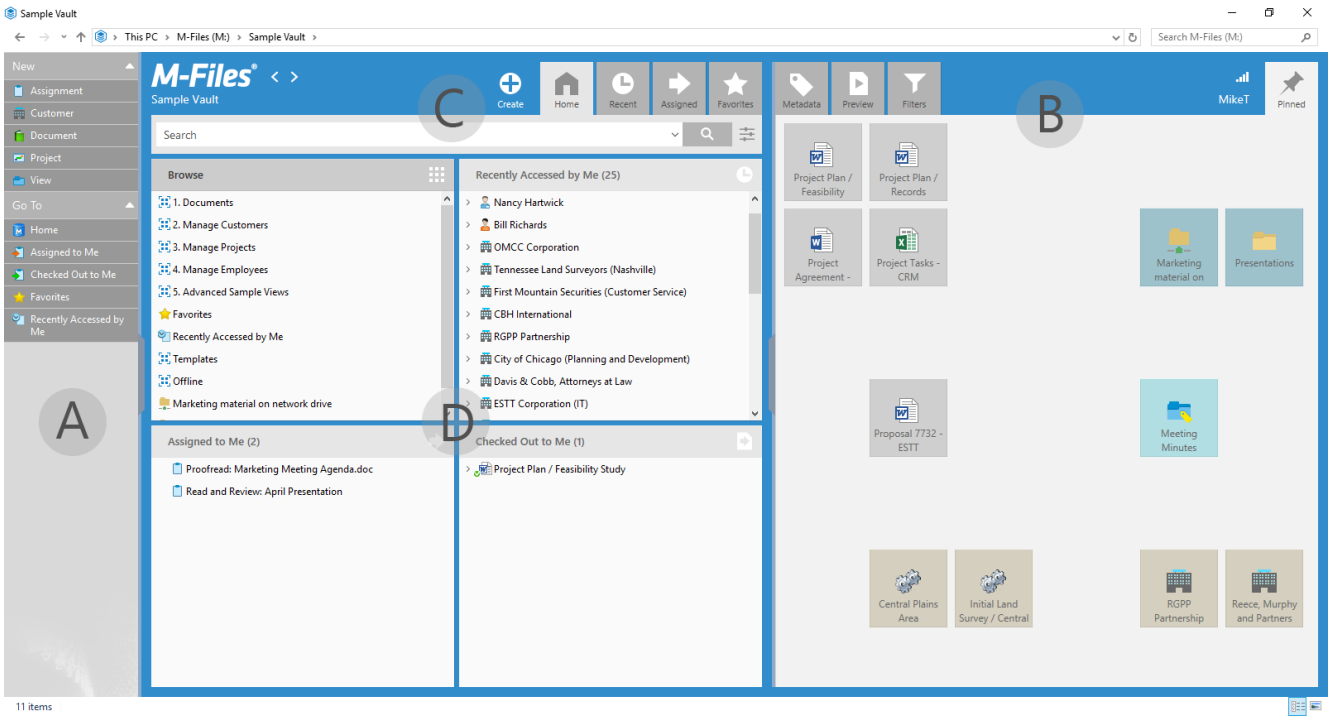


Рисунок 1.2 – Інтерфейс користувача M-Files

Основні характеристики M-Files:

– M-Files використовує метадані для класифікації та ідентифікації документів, що полегшує організацію та пошук інформації;

– система автоматично контролює версії документів, що забезпечує актуальність та точність інформації;

– можливість встановлювати права доступу до конкретних документів чи категорій для забезпечення конфіденційності даних;

– M-Files може інтегруватися з різними системами, такими як Microsoft Office, Outlook, SharePoint, а також з іншими сторонніми застосунками;

– забезпечення електронного підпису для підтвердження автентичності документів та додаткового шару безпеки;

– можливість налаштовувати автоматичні потоки робочих процесів для оптимізації бізнес-процесів;

– можливості пошуку та фільтрації для швидкого та точного доступу до інформації;

– можливість доступу до документів та робочих процесів через мобільні пристрої.

M-Files – це рішення, яке може бути ефективним для організацій, які шукають систему управління документами з акцентом на метадані та ефективність робочих процесів.

1.2.3 M.E.Doc

M.E.Doc – поширене українське програмне забезпечення для подання звітності до контролюючих органів та обміну юридично значущими первинними документами між контрагентами в електронному вигляді. Програма розроблена українською компанією Linkos Group і була створена для автоматизації бухгалтерського обліку та спрощення подання звітності в електронному форматі [6].

Основні характеристики та функціональність M.E.Doc включають:

– облік фінансових операцій, реєстрація податкових обов'язків;

– можливість формування та подання різноманітних звітів, включаючи електронні звіти для податкових органів;

– обробка електронних документів, включаючи формування та обмін електронними рахунками-фактурами;

– можливість інтеграції з іншими електронними системами, зокрема з електронними платіжними системами;

– сприяє автоматизації процесів обробки документів та зменшенню паперової роботи;

– забезпечення безпеки даних та відповідність законодавству про захист персональних даних;

– вбудовані інструменти для аналізу фінансових даних та створення звітності;

– зручний та інтуїтивно зрозумілий інтерфейс для користувачів.

Приклад відображення документів у M.E.Doc наведено на рисунку 1.3.

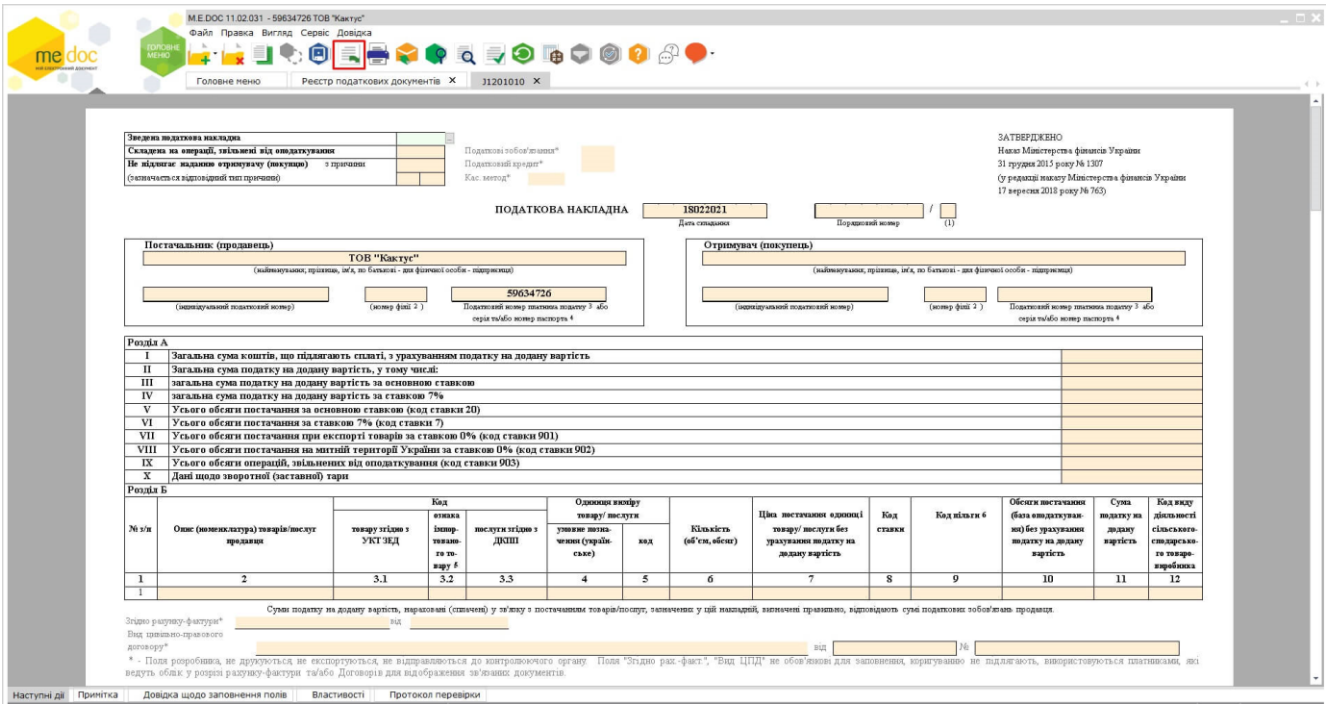


Рисунок 1.3 – Відображення документів у M.E.Doc

M.E.Doc є популярним рішенням в Україні, особливо серед бухгалтерських служб підприємств та бухгалтерів, оскільки вона спрощує процеси бухгалтерського обліку та допомагає виконувати обов'язкові звітність електронним шляхом.

1.2.4 Висновки

Аналіз застосування технології документообігу в існуючих системах дозволяє визначити переваги та недоліки в їхньому використанні.

Переваги:

– збереження часу та ресурсів. Впровадження електронної системи документообігу може значно зменшити час, який витрачається на ручне створення, обробку та збереження паперових документів;

– зручний доступ до інформації. Електронна система дозволяє легко та швидко знаходити потрібну інформацію, що покращує ефективність роботи та сприяє швидшим прийняттям рішень;

– підвищення безпеки і конфіденційності. Інтеграція систем авторизації та аутентифікації забезпечує безпеку даних і може обмежувати доступ до конфіденційної інформації лише авторизованим користувачам;

– легка аналітика та звітність. Системи документообігу можуть забезпечувати аналітичні інструменти та генерацію звітів, що полегшує аналіз ефективності та планування;

– покращення комунікації. Засоби спілкування та обміну повідомленнями в системах документообігу можуть поліпшити внутрішню та зовнішню комунікацію.

Недоліки:

– витрати на впровадження. Впровадження нової системи може вимагати значних витрат на придбання програмного забезпечення, навчання персоналу та інтеграцію з існуючими процесами;

– опір з боку персоналу. Перехід від традиційних паперових систем до електронних може викликати опір серед працівників, які можуть відчувати необхідність у додатковому навчанні та адаптації;

– загрози кібербезпеки. Електронні системи можуть бути піддатливі кібератакам, тому важливо вживати заходів забезпечення кібербезпеки;

– необхідність ретельного обслуговування та оновлення. Електронні системи потребують регулярного обслуговування та оновлення для забезпечення їхньої ефективності та відповідності вимогам;

– неоднорідність інтеграції. У деяких випадках може виникнути складність інтеграції електронних систем документообігу з існуючими інформаційними системами в організації.

Як висновок можна сказати, що застосування технології документообігу в системах приносить численні переваги, проте успішна імплементація вимагає уважного аналізу та управління викликами, які можуть виникнути в процесі змін.

2 ПОСТАНОВКА ЗАДАЧІ НА ДОСЛІДЖЕННЯ

Об’єкт дослідження – впровадження інформаційної технології електронного документообігу у ветеринарних клініках.

Метою дослідження є оцінка та визначення можливостей впровадження технології електронного документообігу в контексті ветеринарної клініки. Основний акцент робиться на покращенні ефективності обробки документів, забезпеченні безпеки та конфіденційності медичних даних, а також спрощенні робочих процесів для підвищення загальної продуктивності клініки.

Задачами ветеринарної клініки є:

– онлайн запис на прийом;
– ведення електронної картки тварин;
– оформлення направлень;
– ведення обліку медичних препаратів та матеріалів;
– управління лабороторними дослідженнями;
– електронна обробка результатів діагностичних процедур.

Для проєктування інформаційної технології документообігу необхідно вирішити низку підпорядкованих завдань, а саме:

– виділити функціональні підсистеми інформаційної системи ветклініки;
– описати зовнішні потоки вхідної та вихідної інформації підсистем та потоки інформації всередині підсистем;
– визначити процеси та дані, з яких формуються документи;
– обгрунтувати вибір методів та засобів перетворення інформації у процесах формування документів.

Для апробації розробленої інформаційної технології електронного документообігу у ветеринарних клініках необхідно розробити серверну (Back-end) та клієнтську (Front-end) частини застосунку, а також мобільний застосунок. Back-end частина програмної системи має за завдання забезпечити ефективну взаємодію

між різними клієнтами. Вона повинна бути програмно реалізована з урахуванням різних шарів, таких як шар представлення, шар доступу до даних та шар сервісів.

Front-end частина програмної системи відповідає за інтерфейс користувача. Вона повинна бути чітко відокремленою від серверної частини, підтримувати роботу клієнтів у вебрежимі та забезпечувати можливість адміністрування системи. Це означає, що вона повинна бути самостійною, але спрямованою на ефективну взаємодію з серверною частиною системи.

3 МОДЕЛЮВАННЯ ПРОЦЕСІВ ОБРОБКИ ІНФОРМАЦІЇ

3.1 Розробка функціональних вимог до інформаційної технології документообігу

Для визначення функціональних вимог до інформаційної технології електронного документообігу ветеринарної клініки було використано метод функціонального моделювання з використанням стандарту IDEF0 (Integration Definition for Function Modeling). Цей підхід дозволив детально розглянути та узгодити ключові етапи та функції системи, визначити вхідні та вихідні дані для формування документів, що сприяє точнішому визначенню вимог та оптимізації процесів електронного документообігу у ветеринарному закладі.

IDEF0 – це методологія функціонального моделювання, спрямована на створення детальних функціональних моделей, що відображають структуру та функції системи, а також потоки інформації і матеріальних об'єктів, які з'єднують ці функції. [7].

Контекстна діаграма A-0, на якій наведена задача «Формувати документи у ветеринарному закладі», зображена на рисунку 3.1. Вона відображає погляд на функції системи з точки зору адміністратора ветеринарного закладу. Було виявлено такі вхідні дані:

- дані користувача;
- дані власника тварини;
- дані тварини;
- дані лікаря;
- графік лікаря;
- список послуг;
- результати досліджень;
- історія хвороби тварини;
- список медикаментів;
- список лабороторних досліджень;

– шаблони документів.

Вихідними даними є:

– створений запис у БД;
– оновлена історія хвороби;
– висновок прийому (pdf);
– рахунок-фактура (pdf);
– підтвердження запису;
– результати лаб.досліджень (pdf).

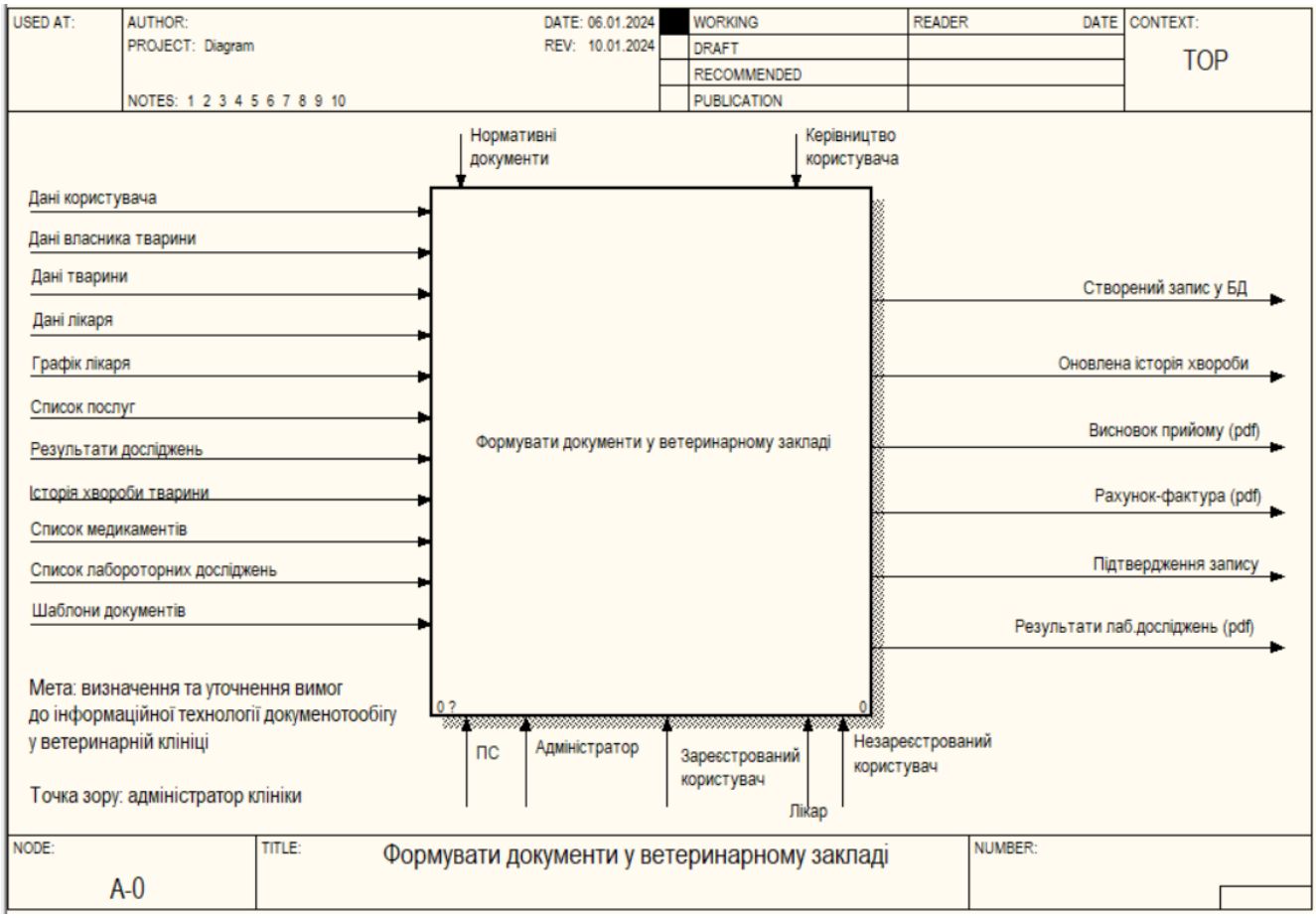


Рисунок 3.1 – Контекстна діаграма системи документообігу ветеринарної клініки

Декомпозиція діаграми А-0 зображена на рисунку 3.2.

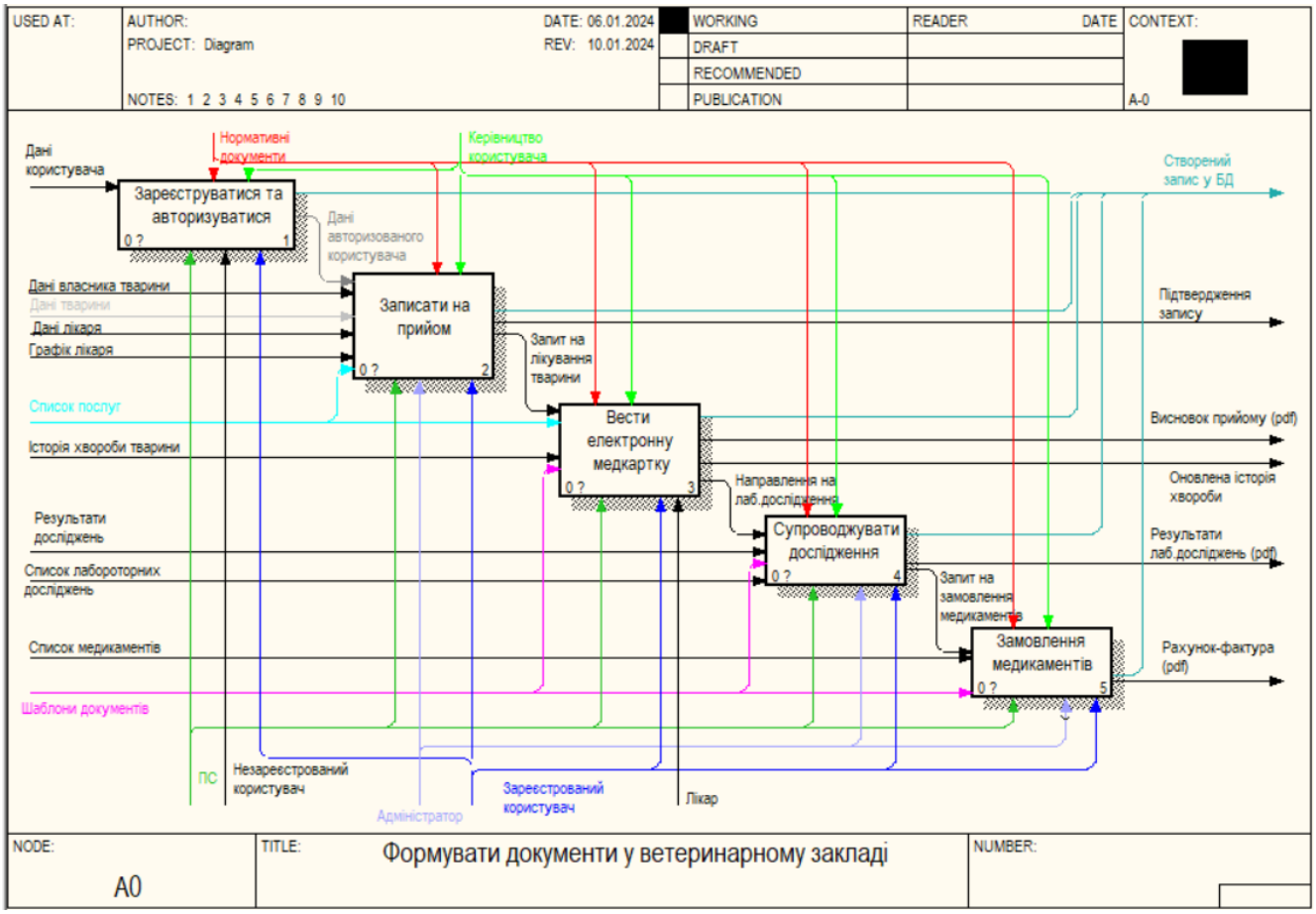


Рисунок 3.2 – Діаграма декомпозиції, яка представляє функції системи

У процесі декомпозиції було отримано такі підфункції:

– зареєстуватися та авторизуватися;
– записати на прийом;
– вести електронну медкартку;
– супроводжувати дослідження;
– замовлення медикаментів.

Функція «Зареєструватися та авторизуватися» пов'язана із створенням облікового запису користувача в інформаційній системі та можливістю входити до цієї системи за допомогою введення ідентифікаційної інформації (логін, пароль). Вона потребує для виконання даних користувача. У результаті виконання функції будуть отримані дані авторизованого користувача, що необхідні будуть для виконання інших функцій системи, та створений запис у базі даних. Декомпозиція цієї функції зображена на рисунку 3.3.

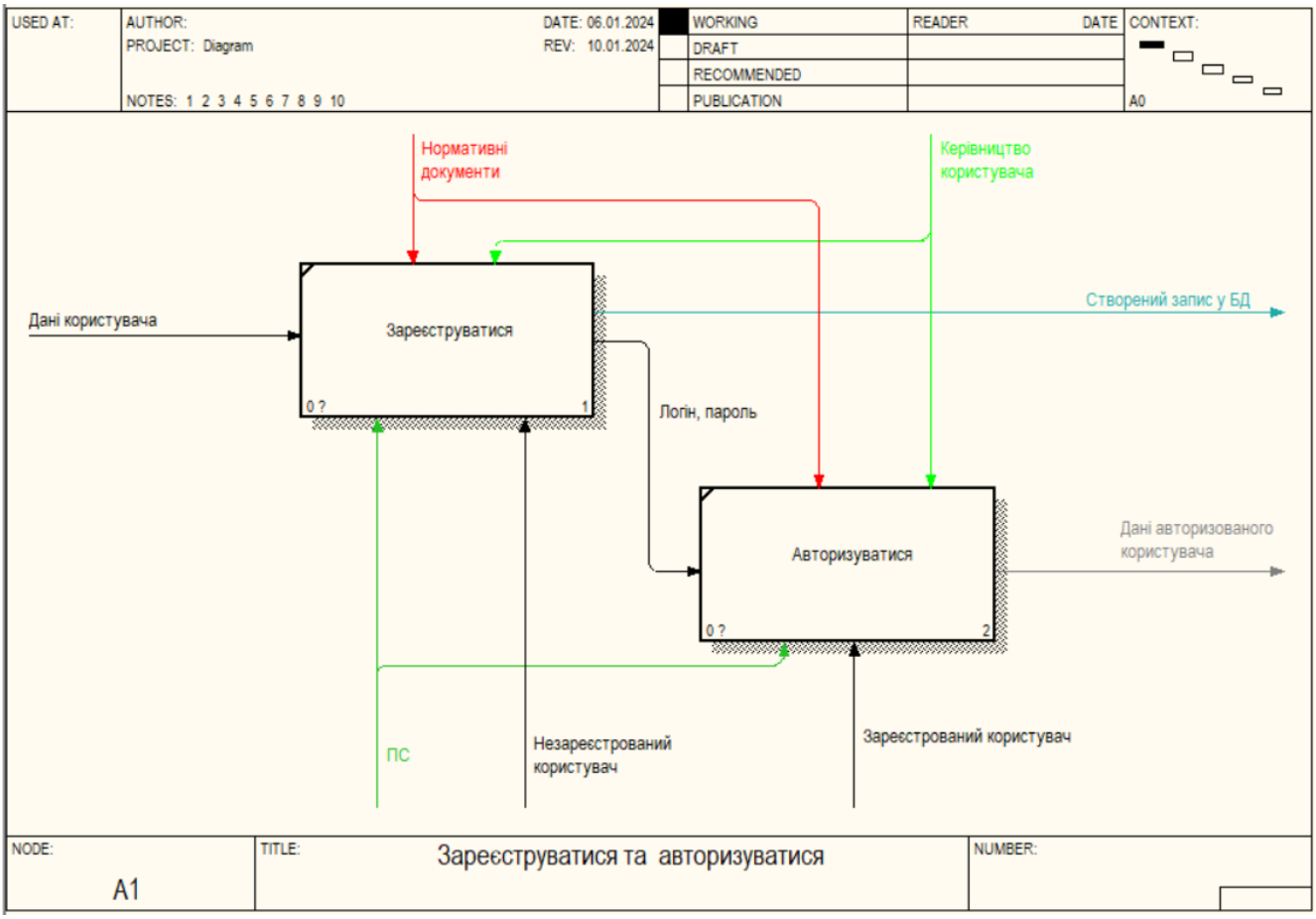


Рисунок 3.3 – Декомпозиція функції «Зареєструватися та авторизуватися»

Функція «Записати на прийом» необхідна для оптимізації процесу та поліпшення зручності для клієнтів та персоналу. Здійснити запис на прийом може лише авторизований користувач. Вхідні дані для функції «Записати на прийом»:

– дані авторизованого користувача;
– дані власника тварини;
– дані тварини;
– дані лікаря;
– графік лікаря;
– список послуг.

Вихідними даними для функції «Записати на прийом» є створений запис у базі даних, підтвердження запису, запит на лікування тварини.

На рисунку 3.4 зображено декомпозиції функції «Записати на прийом», у результаті якої було отримано такі підфункції:

– обрати тварину;
– обрати послугу та лікаря;
– обрати дату та час;
– підтвердити запис.

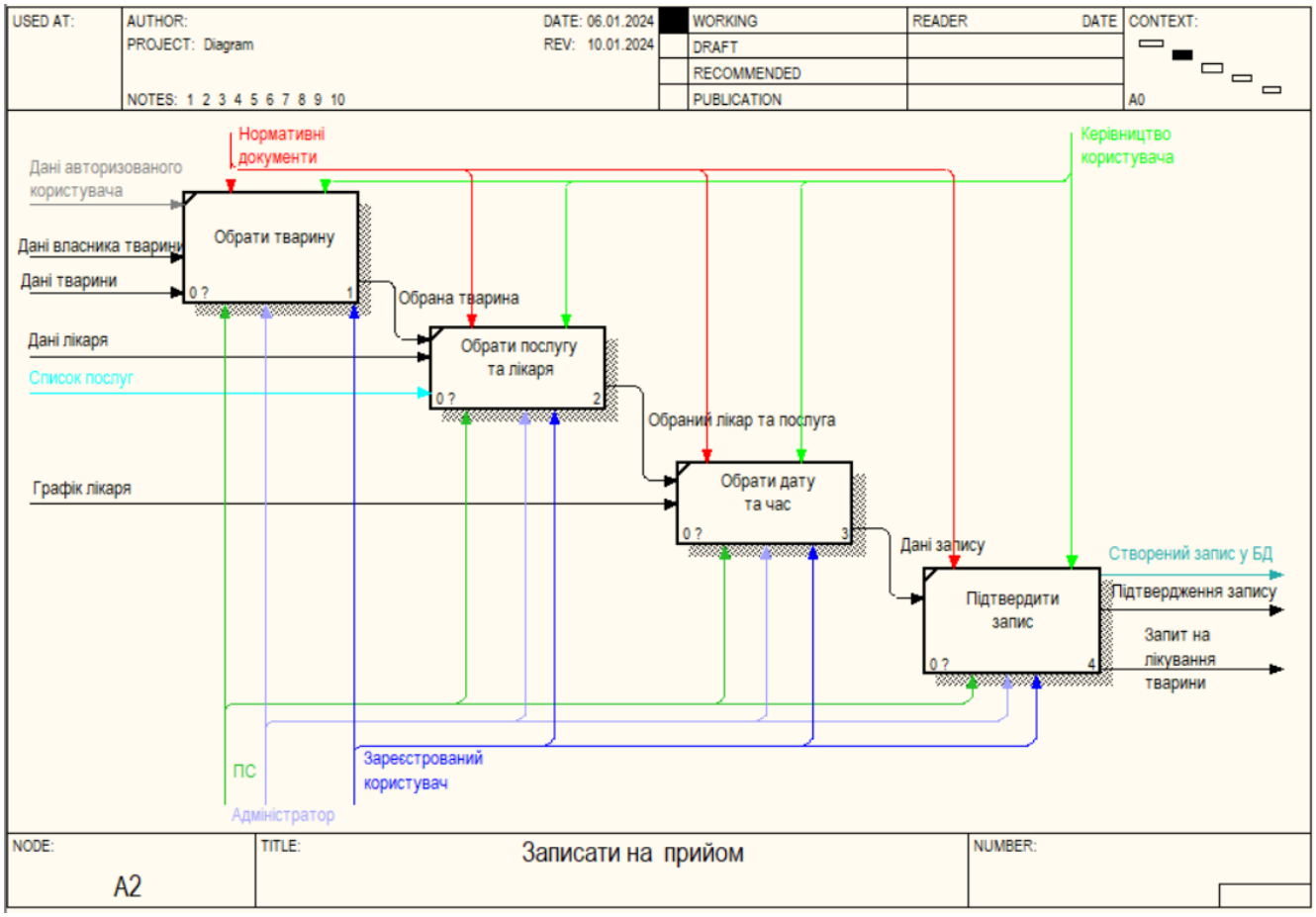


Рисунок 3.4 – Декомпозиція функції «Записати на прийом»

Впровадження функції «Вести електронну медкартку» забезпечує зручність, точність та доступність медичної інформації ветеринарної клініки. Вхідні дані до функції включають запит на лікування тварини, історію хвороби тварини, список послуг, шаблони документів. Вихідними даними до задачі є створений запис у базі даних, сформований документ з висновками прийому у форматі pdf, оновлена історія хвороби, направлення на лабороторні дослідження.

На рисунку 3.5 відображено декомпозицію функції «Вести електронну медкартку». Ця функція була поділена на такі підфункції:

– почати прийом;
– ввести дані у медкартку;
– виписати направлення;
– завершити прийом.

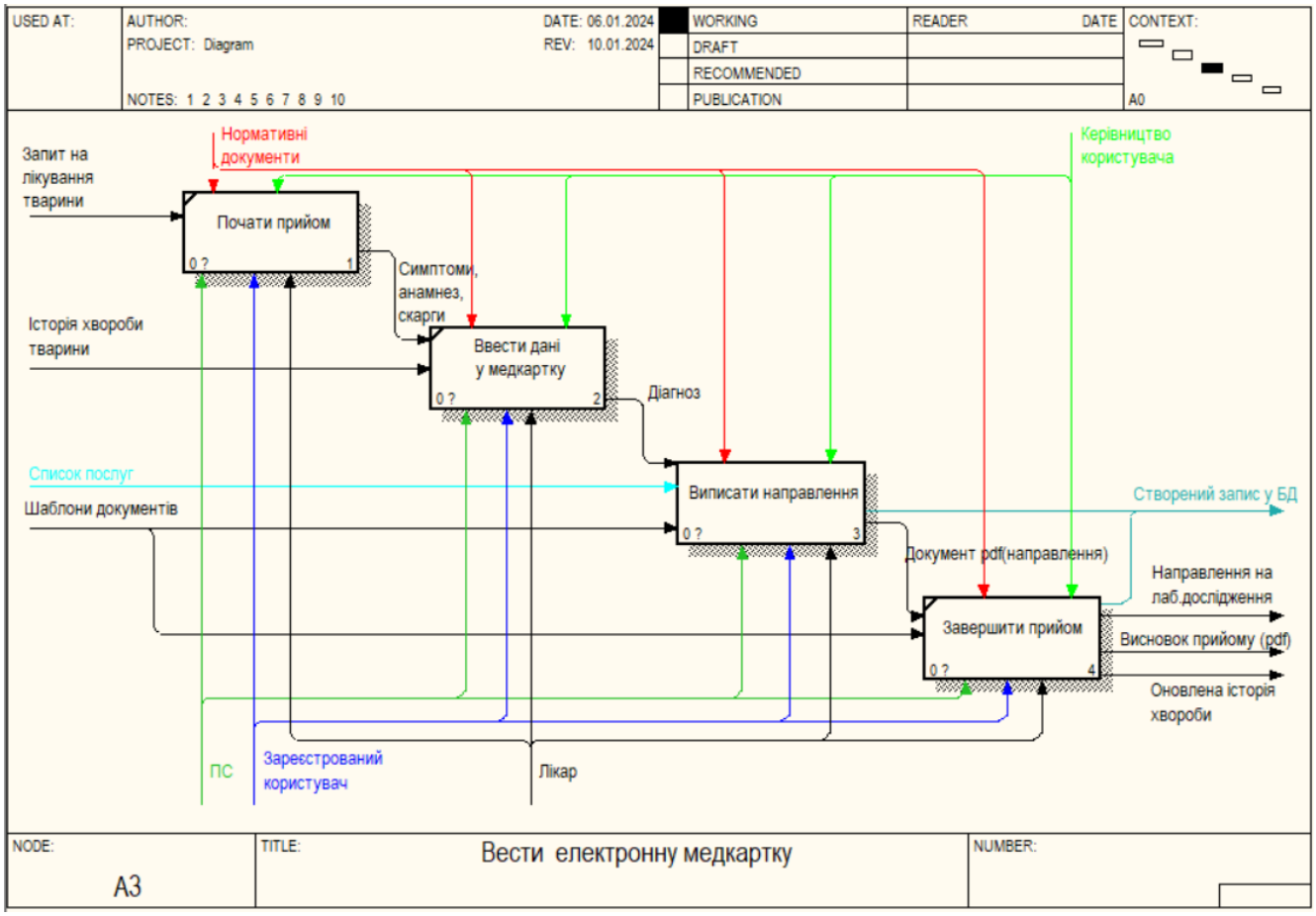


Рисунок 3.5 – Декомпозиція функції «Вести електронну медкартку»

Створення ефективної системи оформлення направлень для лабораторних досліджень та інших консультацій ветеринарної клініки має на меті забезпечення оперативності та точності процесу. Вхідні дані для функції «Виписати направлення»: список послуг, діагноз, шаблони документів. Вихідними даними до функції «Виписати направлення» є створений запис у базі даних, сформоване направлення у форматі pdf.

Декомпозицію функції «Виписати направлення» зображено на рисунку 3.6. Її розбито на такі підфункції:

– обрати послугу;

– створити електронне направлення;

– сформувати документ pdf.

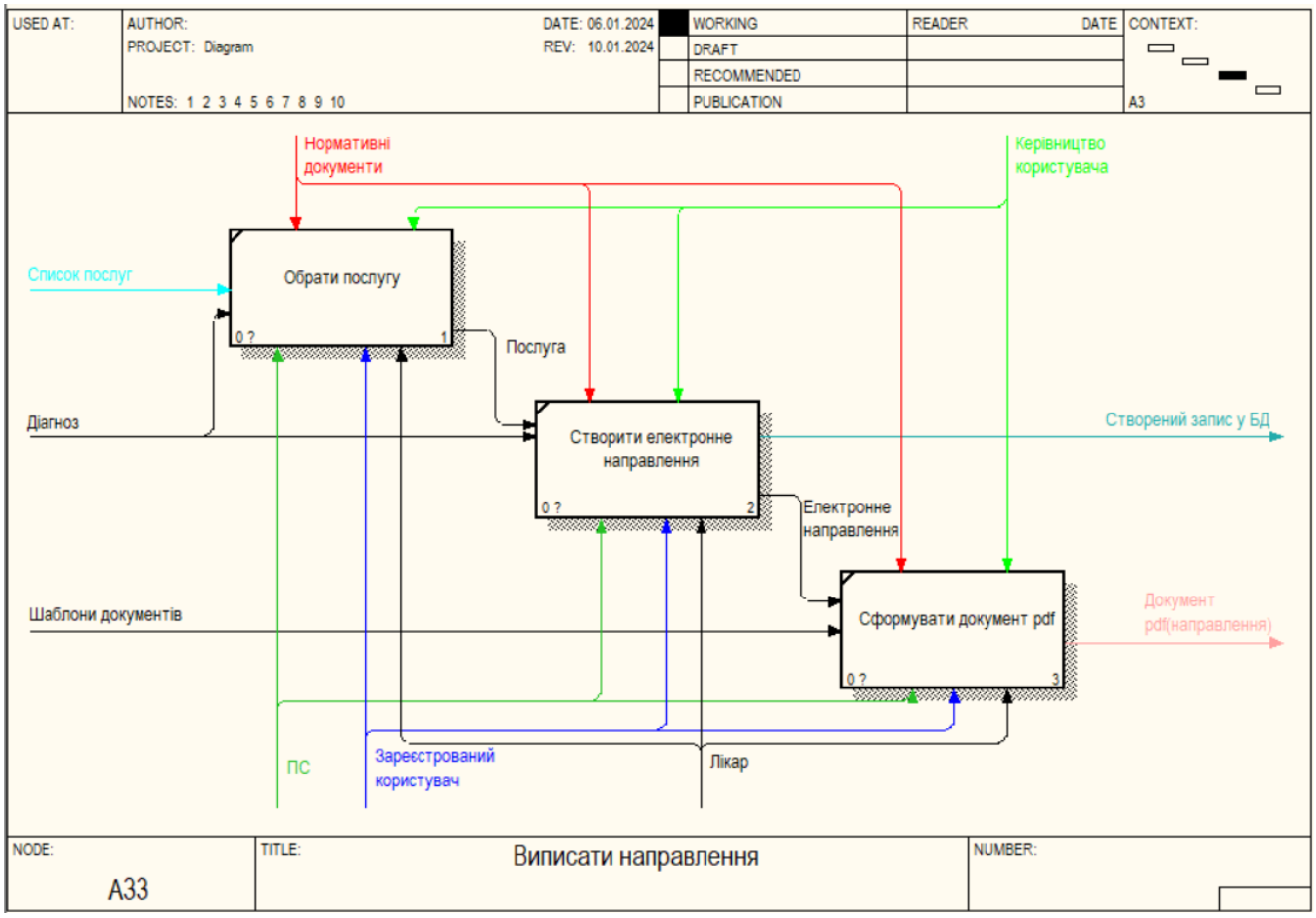


Рисунок 3.6 – Декомпозиція функції «Виписати направлення»

Функція «Супроводжувати дослідження» має такі вхідні дані:

– направлення на лабороторні дослідження;

– результати досліджень;

– список лабороторних досліджень;

– шаблони документів.

Вихідними даними функції є створений запис у БД, результати досліджень у форматі pdf, запит на замовлення медикаментів.

У результаті декомпозиції, яку зображено на рисунку 3.7, було виділено такі підфункції функції «Супроводжувати досідження»:

– створити «Результати досліджень» та веи результати;

– сформувати висновок;

– перевірити наявність медикаментів та засобів.

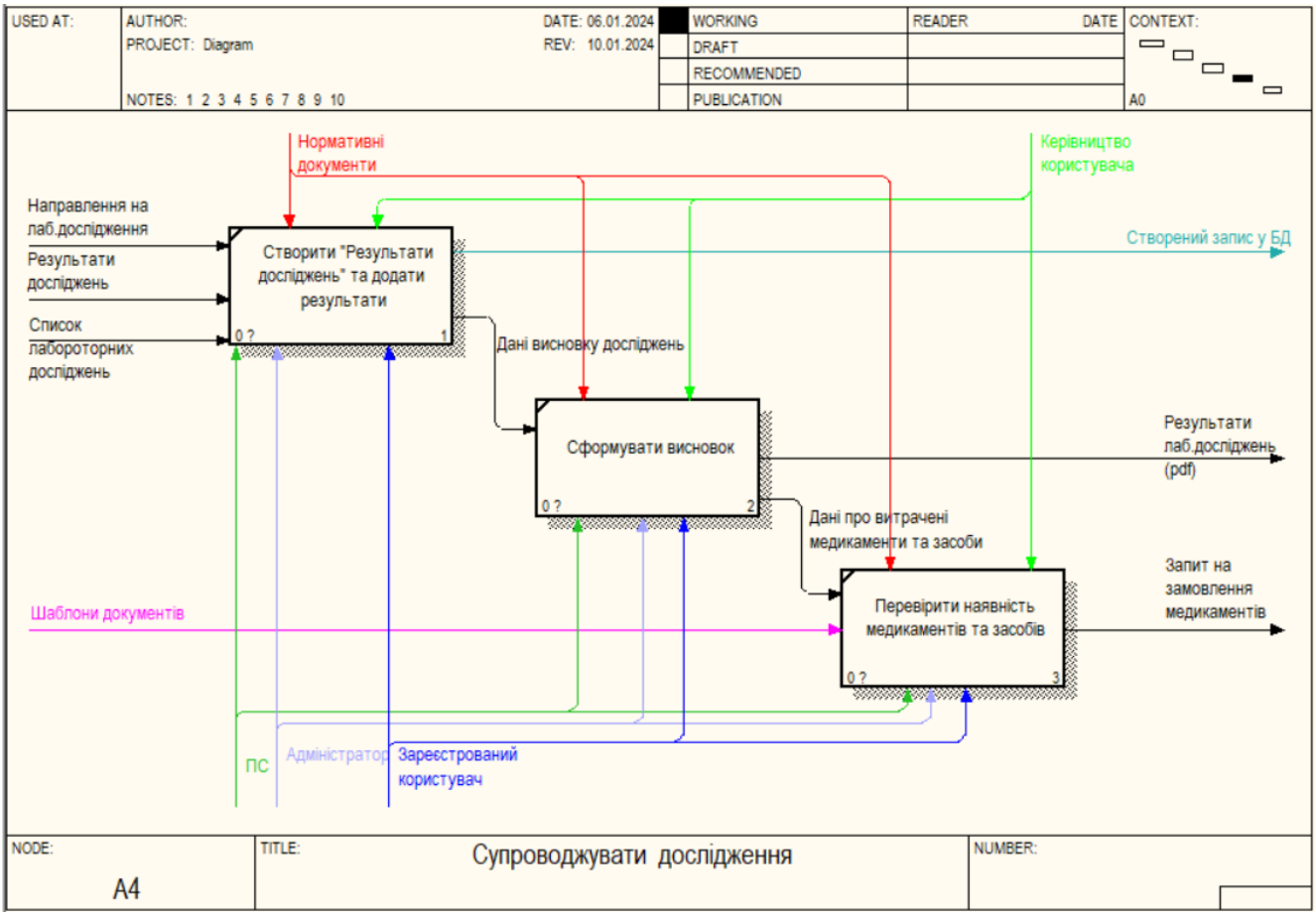


Рисунок 3.7 – Декомпозиція функції «Супроводжувати дослідження»

Функція «Замовлення медикаментів» має такі вхідні дані:

– запит на замовлення медикаментів;

– список медикаментів;

– шаблони документів.

Вихідними даними функції є створений запис у БД, рахунок-фактура у форматі pdf.

На рисунку 3.8 зображено декомпозиції функції «Замовлення медикаментів», у результаті якої було отримано такі підфункції:

– обрати постачальника;

– додати інформацію про медикаменти;

– сформувати рахунок-фактуру.

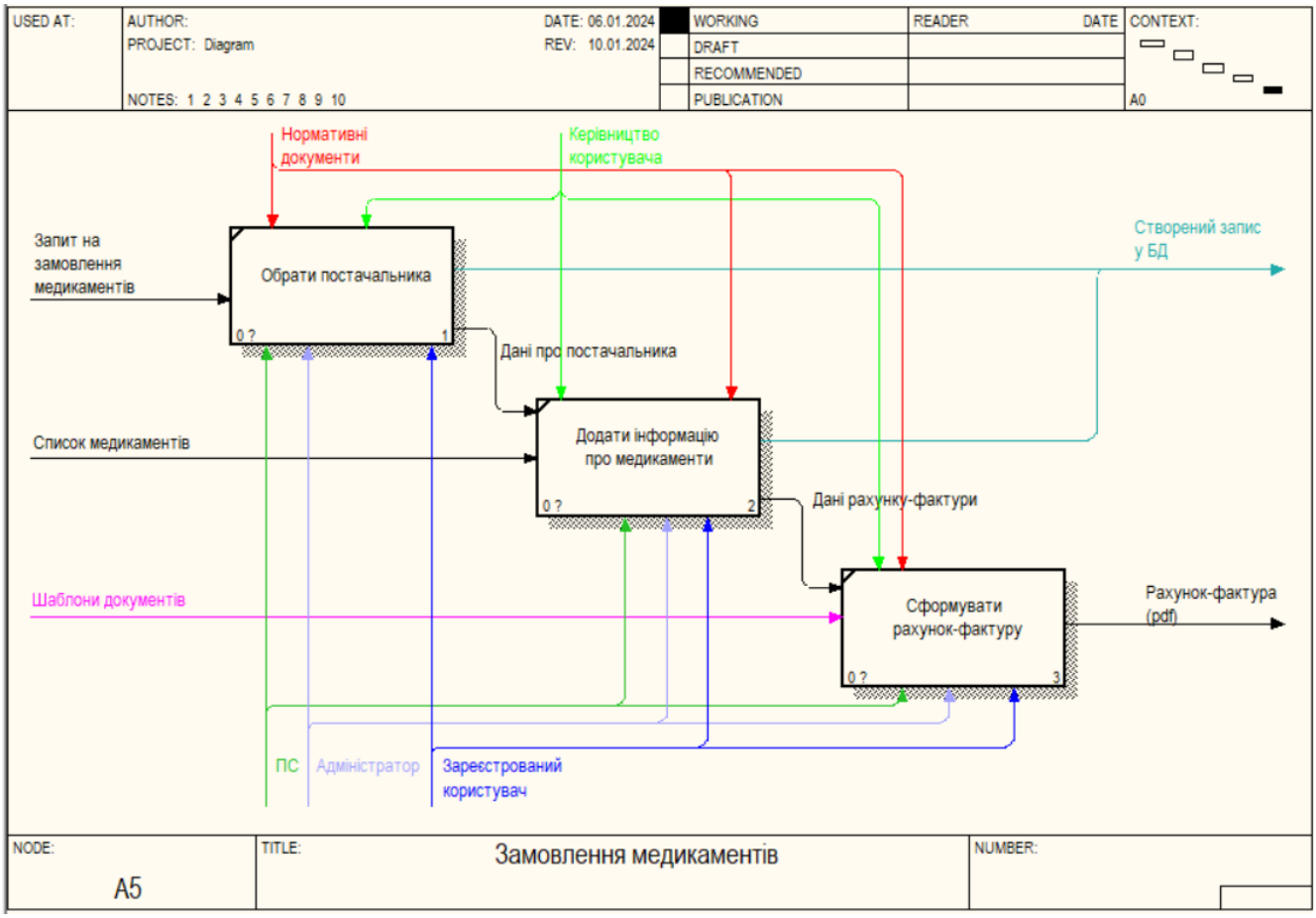


Рисунок 3.8 – Декомпозиція функції «Замовлення медикаментів»

У рамках проведення функціонального моделювання були уточнені функціональні вимоги до розроблювальної інформаційної технології, визначено вхідні та вихідні дані кожної функції, визначено керуючі елементи та нормативні дані, які регулюють та нормують виконання функції, та ресурси чи засоби, які задіяні при виконанні функції.

3.2 Розробка моделі потоків даних інформаційної технології документообігу

Діаграми потоків даних (DFD) – це графічний метод моделювання системи, що використовується для відображення потоків даних та обробки даних всередині системи. [8].

Основна ідея DFD полягає в тому, щоб визначити, як дані переміщуються через систему та як вони обробляються на різних етапах. DFD може бути різним за рівнем деталізації, починаючи від високорівневих оглядових діаграм до деталізованих, що відображають конкретні процеси та дані.

На рисунку 3.9 зображено діаграму потоків даних функції «Зареєструватися та авторизуватися».

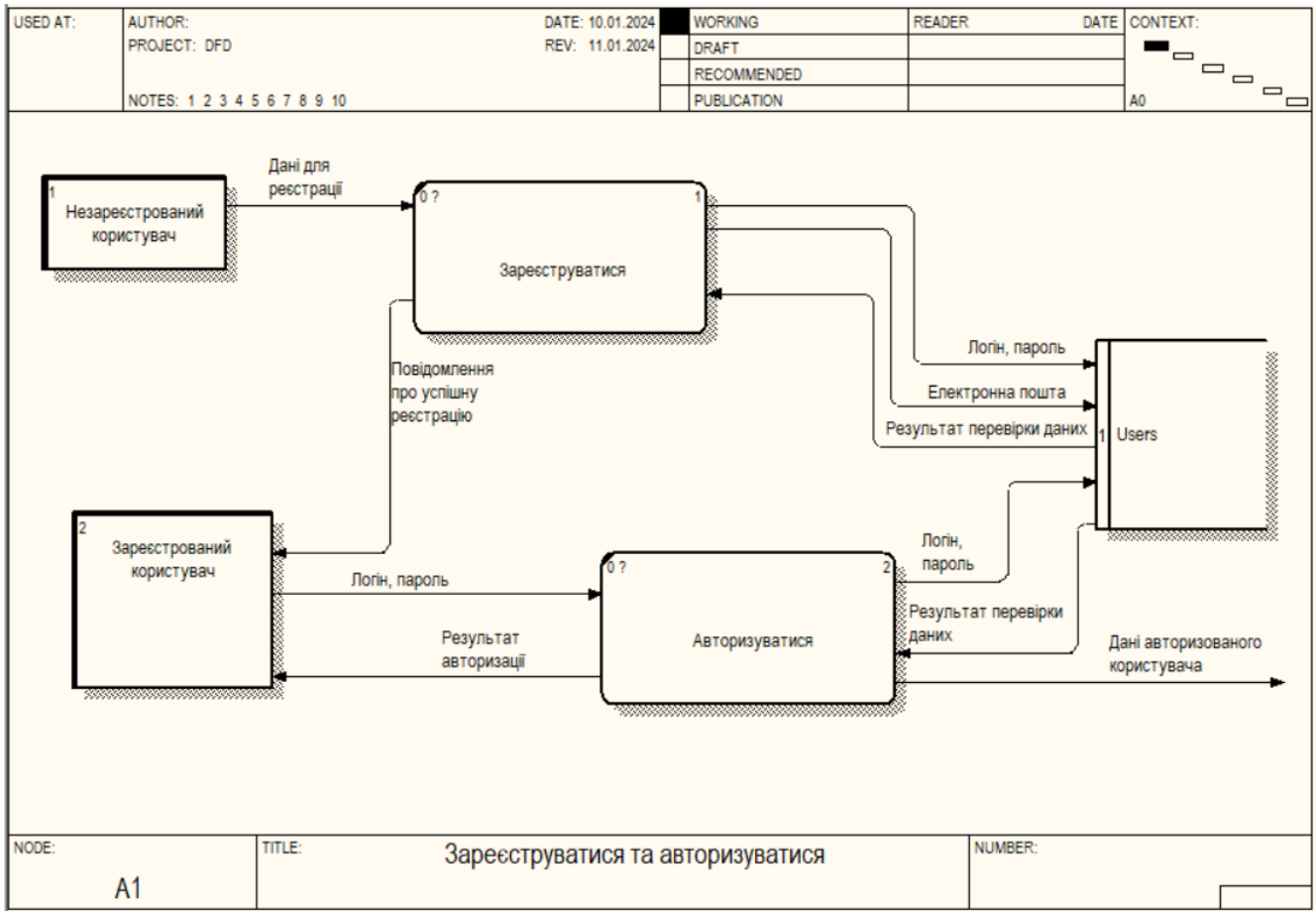


Рисунок 3.9 – Діаграма потоків даних функції «Зареєструватися та авторизуватися»

Для виконання функції «Зареєструватися та авторизуватися» необхідне сховище даних Users, яке дає можливість зберігати особисті дані зареєстрованого користувача, такі як логін, пароль, електронна пошта, перевіряти правильність даних та ідентифікувати користувача при вході у систему.

На рисунку 3.10 представлено діаграму потоків даних функції «Записати на прийом».

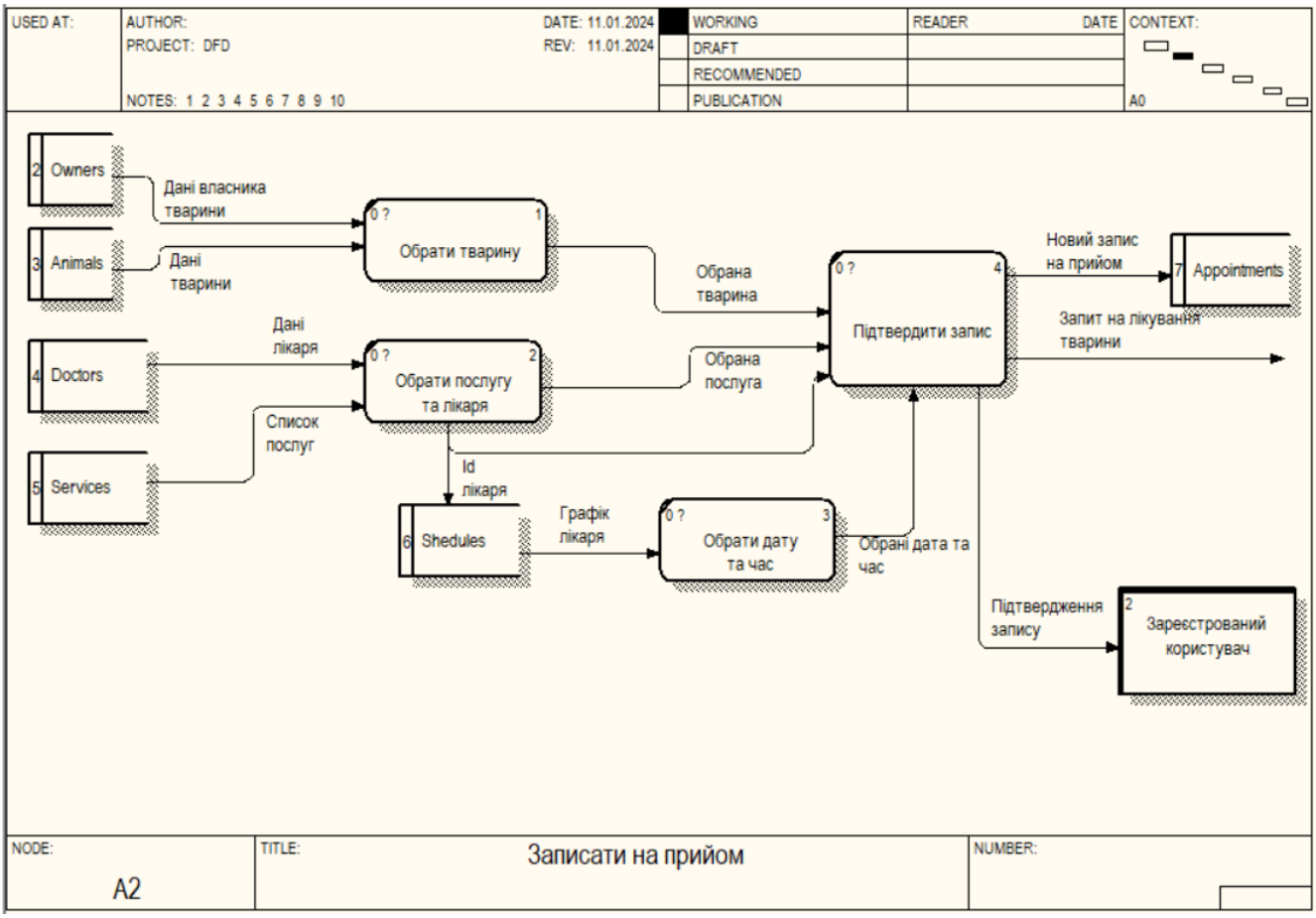


Рисунок 3.10 – Діаграма потоків даних функції «Записати на прийом»

Для виконання функції «Записати на прийом» необхідні такі сховища даних:

– Owners – дає можливість отримувати інформацію про власників тварин;

– Animals – дає можливість отримувати інформацію про тварину;

– Doctors – дає можливість отримувати інформацію про лікарів ветеринарного закладу;

– Services – дає можливість отримувати інформацію про список послуг клініки;

– Shedules – надає інформацію про розклад роботи лікаря;

– Appointments – надає можливість додавати та зберігати створені записи на прийом.

На рисунку 3.11 представлено діаграму потоків даних функції «Вести електронну медкартку».

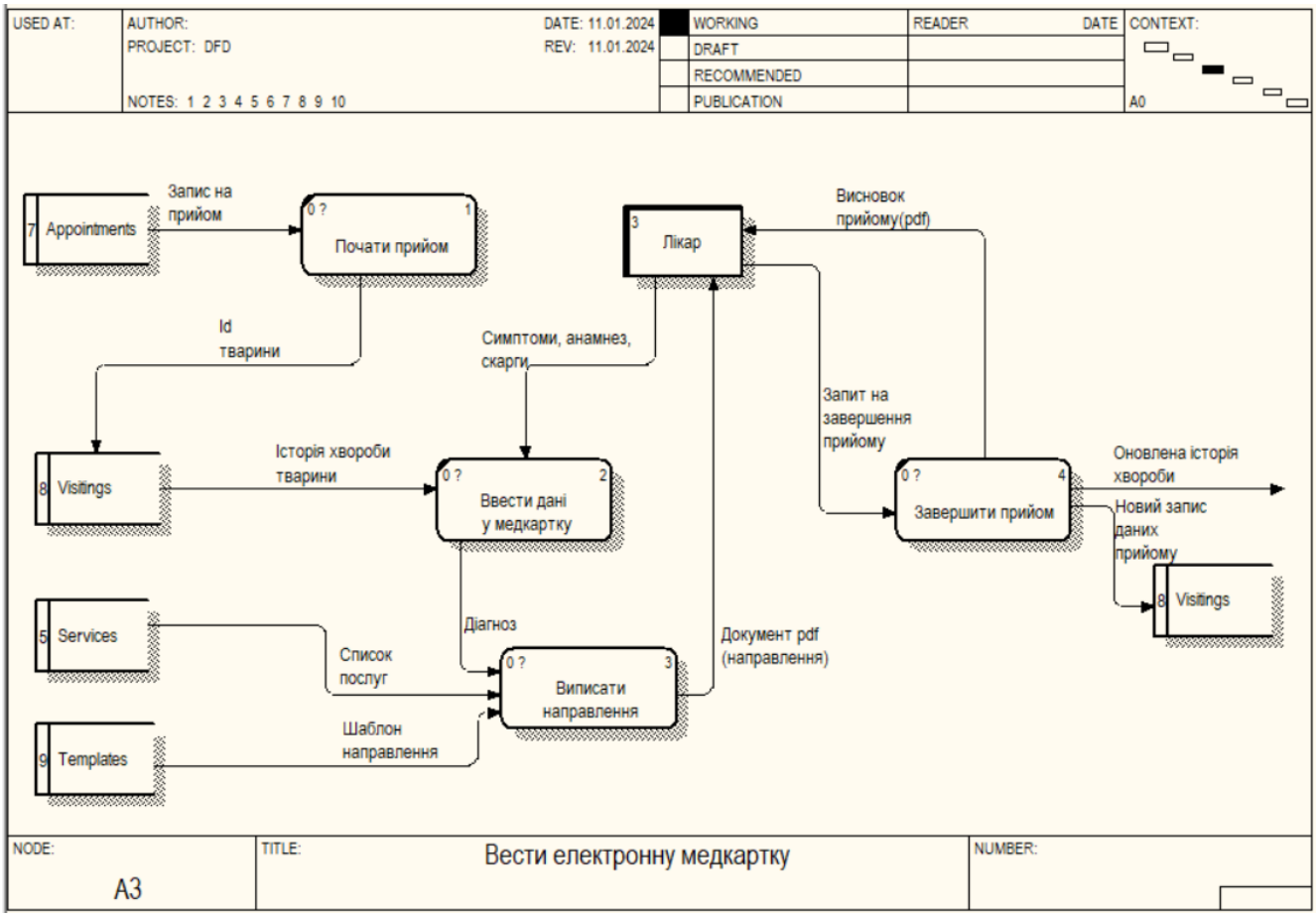


Рисунок 3.11 – Діаграма потоків даних функції «Вести електронну медкартку»

Для виконання функції «Вести електронну медкартку» необхідні такі сховища даних:

– Appointments – дає можливість отримати дані запису на прийом (дані тварини, лікаря, послуга, дата, час);

– Visitings – дає можливість отримати дані про попередні обстеження тварини, а також зберегти дані про поточне;

– Services – дає можливість отримати інформацію про список послуг клініки;

Templates – надає можливість використати шаблон необхідного документу.

На рисунку 3.12 представлено декомпозицію функції «Виписати направлення».

Для виконання функції «Виписати направлення» необхідні такі сховища даних:

– Services – дає можливість отримати інформацію про список послуг клініки;

– Directions – дає можливість зберегти створене направлення.

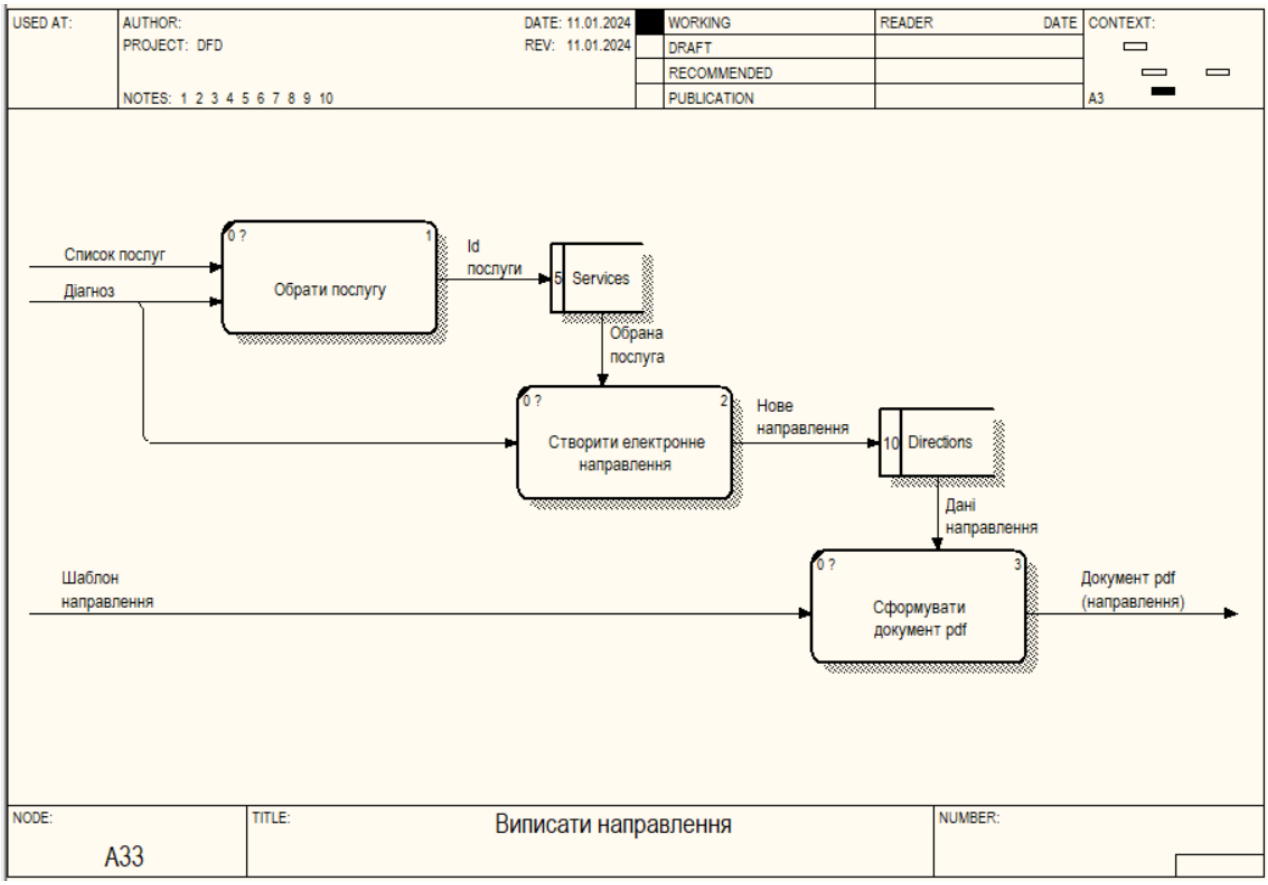


Рисунок 3.12 – Діаграма потоків даних функції «Виписати направлення»

На рисунку 3.13 представлено діаграму потоків даних функції «Супроводжувати дослідження».

Для виконання функції «Супроводжувати дослідження» необхідні такі сховища даних:

– Directions – дає можливість отримати дані напрвлення;

– Research – дає можливість отримати дані про дослідження клініки;

– Research Results – дає можливість отримати результати проведених досліджень;

– Templates – надає можливість використати шаблон необхідного документу;

– Medicines – дає можливість отримати дані про медикаменти та засоби в клініці;

– Spent Medicines – дає можливість отримати інформацію про витрачені медикаменти під час досліджень;

– Necessary Medicines – дає можливість зберегти дані про медикаменти та засоби, які необхідно замовити.

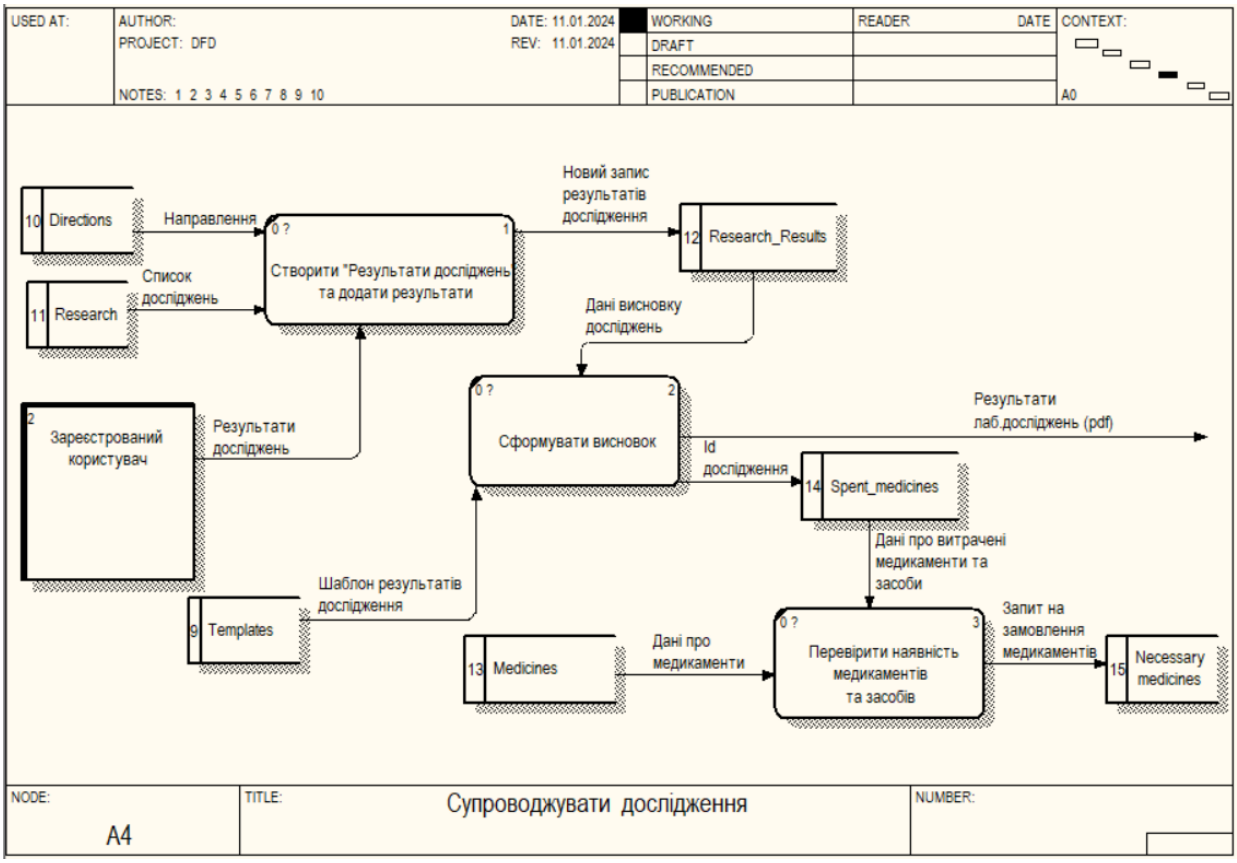


Рисунок 3.13 – Діаграма потоків даних функції «Супроводжувати дослідження»

На рисунку 3.14 представлено діаграму потоків даних функції «Замовлення медикаментів».

Для виконання функції «Замовлення медикаментів» необхідні такі сховища даних:

– Suppliers – дає можливість отримати дані про постачальників;

– Necessary Medicines – дає можливість зберегти дані про медикаменти та засоби, які необхідно замовити;

– Invoices – дає можливість зберегти рахунок-фактуру;

– Templates – надає можливість використати шаблон необхідного документу;

– Medicines – дає можливість отримати дані про медикаменти та засоби в клініці;

– Medicine Invoice – дає можливість зберегти необхідні медикаменти та засоби до рахунку-фактури.

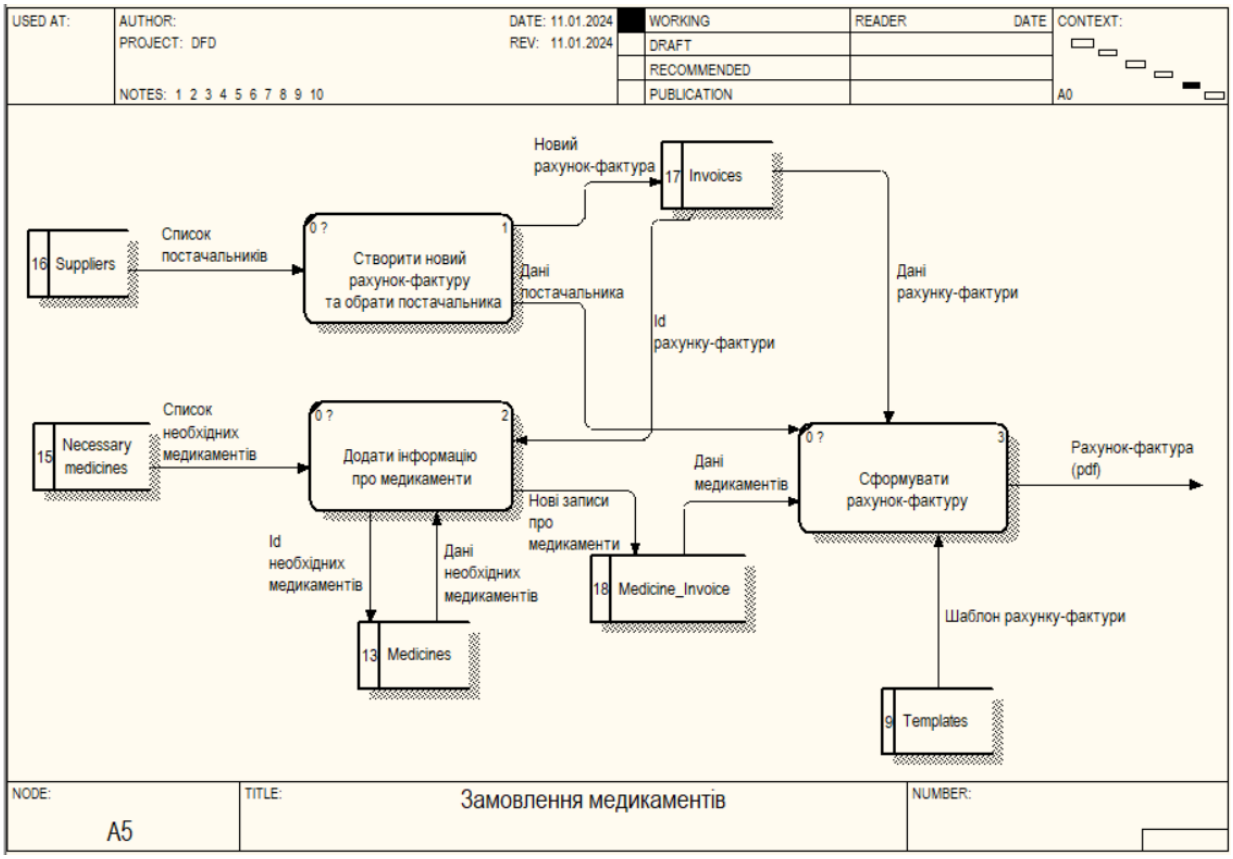


Рисунок 3.14 – Діаграма потоків даних функції «Замовлення медикаментів»

Сукупність діаграм потоків даних визначає сховища даних, потоки інформації всередині системи, місця виникнення документів, їх користувачів, і дає цілісне уявлення про документообіг в цілому у ветзакладі.

3.3 Логічне моделювання даних

Для розробки інформаційної технології документообігу ветеринарної клініки необхідно розробити базу даних, де будуть зберігатися дані клініки.

Логічна структура бази даних наведена на рисунку 3.15.

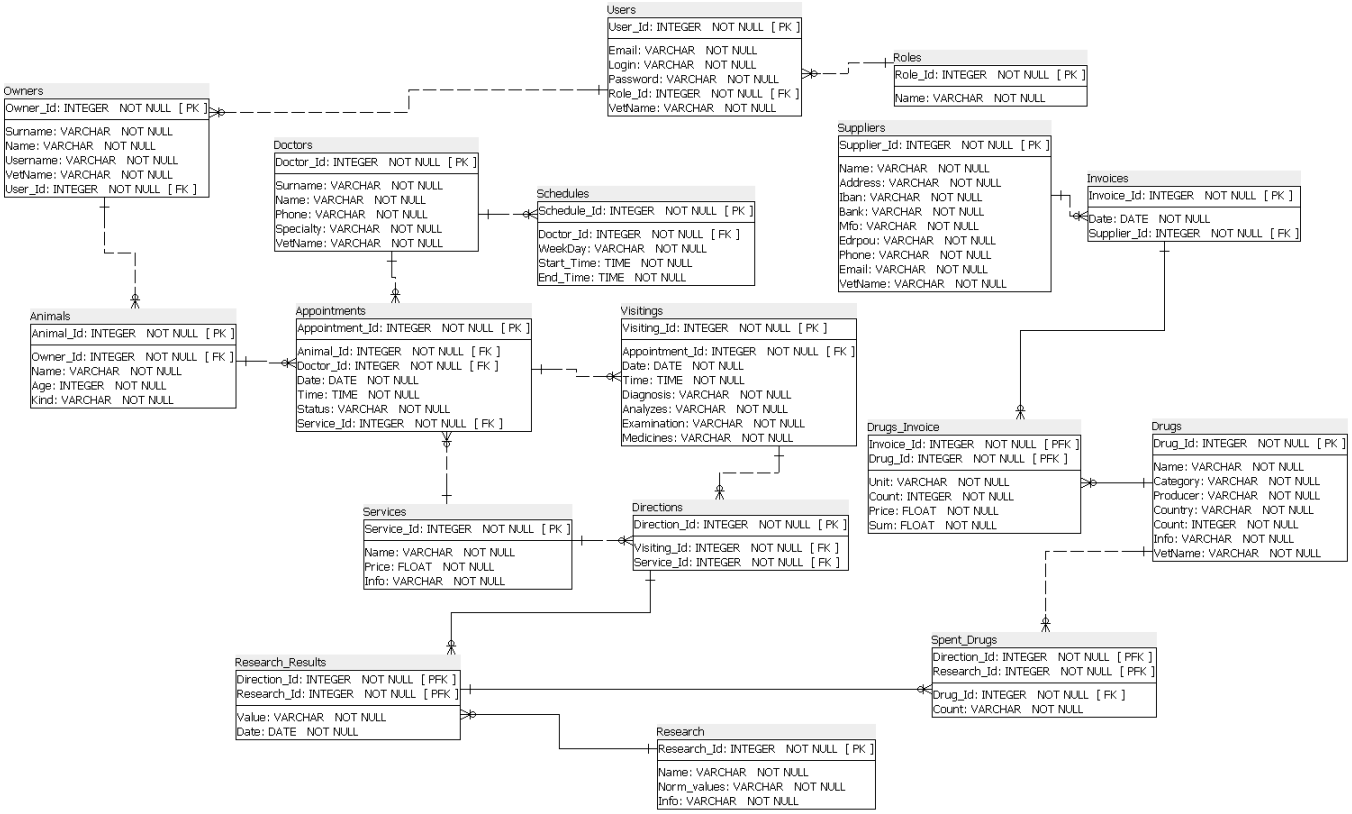


Рисунок 3.15 – Логічна структура бази даних

Розглянемо сутності бази даних:

– Users – зберігає дані користувачів системи(логін, пароль, електронна пошта, роль, назва ветеринарної клініки);

– Roles – ролі користувачів системи;

– Owners – зберігає дані власників тварин, а саме прізвище та ім’я, username, назву ветеринарної клініки;

– Animals – містить дані тварини, такі як кличка, вік та вид тварини. Також у сутності збережений ідентифікатор власника тварини;

– Doctors – зберігає дані лікаря ветеринарного закладу: прізвище та ім’я, спеціальність, мобільний телефон, назву закладу;

– Schedules – містить інформацію про розклад роботи окремого лікаря, а саме ідентифікатор необхідного лікаря, день тижня та час початку та завершення роботи у цей день;

– Appointments – містить в собі інформацію про створені записи на прийом у клініці, а саме ідентифікатори тварини, яку записують на прийом, та лікаря, дату та час, ідентифікатор послуги, на яку здійснюється запис, статус виконання запису.

– Services – містить дані про послуги ветеринарного закладу: назву, ціну, додаткову інформацію про послугу;

– Visitings – зберігає дані кожного прийому тварини у лікаря: ідентифікатор запису на прийом, дату та час проведення прийому, встановлений діагноз, призначені аналізи, обстеження, медикаменти;

– Directions – містить в собі інформацію про направлення: ідентифікатори прийому, на якому було створено направлення і за допомогою якого можна буде отримати дані про тварину, лікаря, та послуги.

– Suppliers – містить в собі дані про постачальників медикаментів та засобів у ветеринарну клініку: назва, адреса, банк, IBAN, МФО, ЕДРПОУ, телефон, електронна пошта, назва ветеринарного закладу;

– Drugs – містить інформацію про медикаменти та засоби у клініці: назва, категорія, виробник, країна, кількість, додаткова інформація, назва ветеринарного закладу;

– Invoices – містить інформацію про рахунки-фактури: ідентифікатор постачальника та дату;

– Drug_Invoice – містить перелік медикаментів та засобів у окремих рахунках-фактурах: ідентифікатори рахунку-фактури та медикаменту, одиницю виміру, кількість, ціну та суму;

– Research – містить дані про лабороторні дослідження, які проводять у закладі: назву, нормативні значення, додаткову інформацію;

– Research_Results – містить у собі інформацію про результати досліджень: ідентифікатори дослідження та направлення, результат, дату;

– Spent_Drugs – містить в собі дані про витрачені засоби та медакаменти.

4 ДОСЛІДЖЕННЯ МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

4.1 Математичний опис методів рішення задачі документообігу

4.1.1 Загальний опис

Документообіг – рух документів в організації з моменту їх отримання чи створення до завершення виконання, відправки чи передачі в архів [9].

Процес документообігу може бути представленим у вигляді трьох кінцевих множин і зв’язків елементів цих множин. У формулі 4.1 наведено математичне представлення цього процесу:

$$O = \{D, P, U\} \tag{4.1}$$

де O – кінцева множина відношень, D – множина документів, P – множина етапів обробки документів, U – множина учасників процесу документообігу.

Процес документообігу у ветеринарній клініці можна розглядати як систему взаємодії цих трьох кінцевих множин та зв'язків між їхніми елементами. Розглянемо ці множини:

– множина "Документи". Ця множина включає всі документи, які обробляються в рамках системи документообігу ветеринарної клініки. Елементи: медичні картки тварин, направлення, звіти, результати аналізів, тощо;

– множина "Етапи". Ця множина включає різні етапи або кроки, через які проходять документи під час обробки. Елементи: створення, редагування, архівування, тощо;

– множина "Учасники". Елементи: клієнти (власники тварин), лікарі, медичні асистенти, адміністратори.

Розглянемо зв’язки між множинами:

– документи-учасники. Кожний документ пов’язаний з конкретними учасниками;

– учасники-етапи. Учасники можуть виконувати різні етапи обробки;

– етапи-документи. Документи проходять через різні етапи обробки в рамках системи документообігу, від створення до завершення.

Наведемо конкретний приклад множин та зв’язків між ними:

– реєстрація тварини у системі. Учасник «адміністратор» виконує етапи «створення» та «редагування» для документу «медична картка тварини»;

– медичний огляд. Учасник «ветеринарний лікар» виконує етапи «перегляд» та «редагування» для документу «медична картка тварини», етапи «створення» та «редагування» для документу «консультаційний висновок спеціаліста»;

– виписка направлень. Учасник «ветеринарний лікар» виконує етап «створення» для документу «електронне направлення»;

– проведення діагностичних процедур. Учасник «лікар-діагност» етапи «перегляд» та «редагування» для документу «медична картка тварини», виконує етапи «створення» та «редагування» для документу «результат УЗД»;

– проведення лабороторних досліджень. Учасник «медична сестра» виконує етапи «створення», «редагування», «відправка» для документу «результат аналізів»;

– отримання результатів аналізів. Учасник «власник тварини» виконує етап «перегляд» для документу «результат аналізів»;

повторний медичний огляд. Учасник «ветеринарний лікар» виконує етапи «перегляд» та «редагування» для документу «медична картка тварини», етап «перегляд» для документу «результат УЗД», етап «перегляд» для документу «результат аналізів», етапи «створення» та «редагування» для документу «консультаційний висновок спеціаліста».

4.1.2 Математичний опис задачі генерації файлів

Для автоматизованої обробки документів важливо, щоб кожен документ мав чітко визначену структуру – шаблон. Цей шаблон є об'єктом, який описує постійні елементи інформації, що містяться в документах, створених за цим шаблоном, а

також поля для змінної інформації. Такий підхід дозволяє описати методи і механізми заповнення цих полів.

Нехай D – множина всіх документів, де d_i – конкретний документ. Нехай T – множина шаблонів, де t_j – конкретний шаблон документа. Нехай R – множина всіх записів у базі даних, де r_k – конкретний запис. Кожен шаблон t_j представляє собою формалізовану структуру, яка містить сталі частини інформації (S_{t_j}) та поля для змінної інформації (P_{t_j}). Визначимо функцію $F(t_j,r_k)$, яка описує методи та механізми заповнення полів змінної інформації в шаблоні t_j з використанням даних з конкретного запису в базі даних r_k:

$$P_{t_j} = F(t_i, r_k) \tag{4.2}$$

Методи і механізми заповнення полів:

– запити до бази даних;
– обчислення на основі існуючих даних;
– генерація унікальних ідентифікаторів;
– заповнення на основі правил.

Для формування вихідного документа d_i за шаблоном t_j визначимо функцію $G(t_j,P_{t_j})$, яка використовує заповнені поля (P_{t_j}) для створення нового документа:

$$d_i = G(t_j, P_{t_j}) \tag{4.3}$$

Таким чином, задача генерації текстових файлів формалізована через функції, які визначають способи заповнення та формування текстових файлів з використанням математичних конструкцій.

4.2 Аналіз методів генерації файлів

Генерація файлів для системи документообігу у ветеринарній клініці важливою частина процесу управління даними та документами. Два основних формати файлів, які використовуються для документів у ветеринарних клініках, це pdf і docx.

4.2.1 Методи генерації файлів у форматі pdf

ItextSharp – це вдосконалена бібліотека інструментів, яка використовується для створення складних звітів у форматі PDF. Itext використовується різними технологіями – розробники Android, .NET, Java і GAE використовують його для вдосконалення своїх програм за допомогою функцій PDF. Він створює документи та звіти на основі даних із баз даних або файлів xml, а також об’єднує або розділяє сторінки з наявних PDF-файлів. [10]

iTextSharp потужна бібліотека для роботи з PDF-документами на мові програмування C#. Вона дозволяє створювати, редагувати, об'єднувати та розділяти PDF-файли, додавати зображення, текст, таблиці, формувати документи, а також здійснювати різні операції з обробки PDF-даних.

Наведемо основні можливості використання цієї бібліотеки [11]:

– створення PDF-документів. Document – основний клас, який представляє PDF-документ. Його можна ініціювати і передати об'єкту PdfWriter для запису даних в PDF-файл;

– додавання елементів. Paragraph, Image, Table – класи для додавання тексту, зображень та таблиць до документу;

– форматування тексту. TextStyle, Paragraph дозволяють змінювати стиль та форматування тексту у документі;

– робота з зображеннями. Image – клас для вставки зображень у документ;

– створення таблиць. Table, Cell дозволяють створювати та редагувати таблиці;

– створення звітів. Canvas, Chart дозволяють вбудовувати графіки та діаграми у PDF-документи;

– обробка подій та водяних знаків. IEventHandler дозволяє обробляти події, такі як додавання водяних знаків або власних елементів.

PDFsharp – це бібліотека .NET з відкритим вихідним кодом, яка легко створює та обробляє PDF-документи за допомогою .NET 6 під Windows, Linux і Mac [10].

Характеристки PDFsharp [12]:

– створює PDF-документи на будь-якій мові .NET;

– проста для розуміння об'єктна модель для складання документів;

– нещодавно розроблений з нуля і повністю написаний на C#;

– один вихідний код для малювання на сторінці PDF, а також у вікні або на принтері;

– зміна, об'єднання та розділення існуючих PDF-файлів;

– зображення з прозорістю (кольорова маска, монохромна маска, альфа-маска);

– вбудовування і підмножина шрифтів

– графічна реалізація на основі GDI+ або WPF.

Наведемо основні можливості використання цієї бібліотеки:

– створення PDF-документів. PdfDocument, PdfPage – основні класи для створення нового PDF-документу та додавання сторінок;

– додавання тексту. XFont, XBrush використовуються для визначення шрифту та кольору тексту;

– додавання зображень. XImage використовується для вставки зображень у документ;

– створення таблиць. XGraphics, XTextFormatter дозволяють створювати та форматувати таблиці;

– робота з розміщенням тексту та форматуванням. XTextFormatter дозволяє встановлювати розміщення тексту та його форматування.

У таблиці 3.1 наведена порівняльна характеристика бібліотек ITextSharp та PdfSharp.

Таблиця 4.1 – Порівняльна характеристика бібліотек ITextSharp та PdfSharp

Характеристика	iTextSharp	PdfSharp
Ліцензія	AGPL, комерційні варіанти	MIT License
Функціональність	Розширені можливості для роботи з PDF	Базовий функціонал для роботи з PDF
Створення PDF	Так	Так
Редагування PDF	Так, редагування, об'єднання, розділення, заповнення форм	Обмежена функціональність стосовно редагування
Додавання тексту	Так	Так
Додавання зображень	Так	Так
Робота з таблицями	Так	Так
Локалізація тексту	Так	Так
Робота з шрифтами	Так	Так
Розміщення тексту	Так	Так
Форматування тексту	Так	Обмежено
Робота з графікою	Так	Обмежено
Спрощеність використання	Має велику кількість функціональності, може бути складніше в освоєнні	Простіша, менше функціональності
Спільнота та підтримка	Активна спільнота та регулярні оновлення	Менш активна спільнота, менше оновлень
Розширені можливості	Великий спектр функціональності	Менше розширених можливостей

iTextSharp є більш розширеною бібліотекою, але вона має обмеження за ліцензією, які можуть впливати на вибір у великих комерційних проектах. PdfSharp зручна для простих сценаріїв, але вона менш розширена за функціональністю.

4.2.2 Методи генерації файлів у форматі docx

Aspose.Words для .NET – це бібліотека класів, яка дозволяє вашим програмам виконувати широкий спектр завдань із обробки документів. Aspose.Words підтримує більшість популярних форматів документів, таких як DOC, DOCX, RTF, HTML, Markdown, PDF, XPS, EPUB та інші. За допомогою Aspose.Words для .NET можна створювати, змінювати, конвертувати, відтворювати та друкувати документи без сторонніх програм або автоматизації офісу [13].

Використання Aspose.Words для .NET у проекті дає такі переваги [13]:

- багатий набір функцій;
- незалежність від платформи;
- незалежність від сторонніх програм;
- продуктивність і масштабованість;
- мінімальна крива навчання.

На діаграмі (рис. 3.1) показано основні функції Aspose.Words для .NET і те, як вони пов’язані між собою.

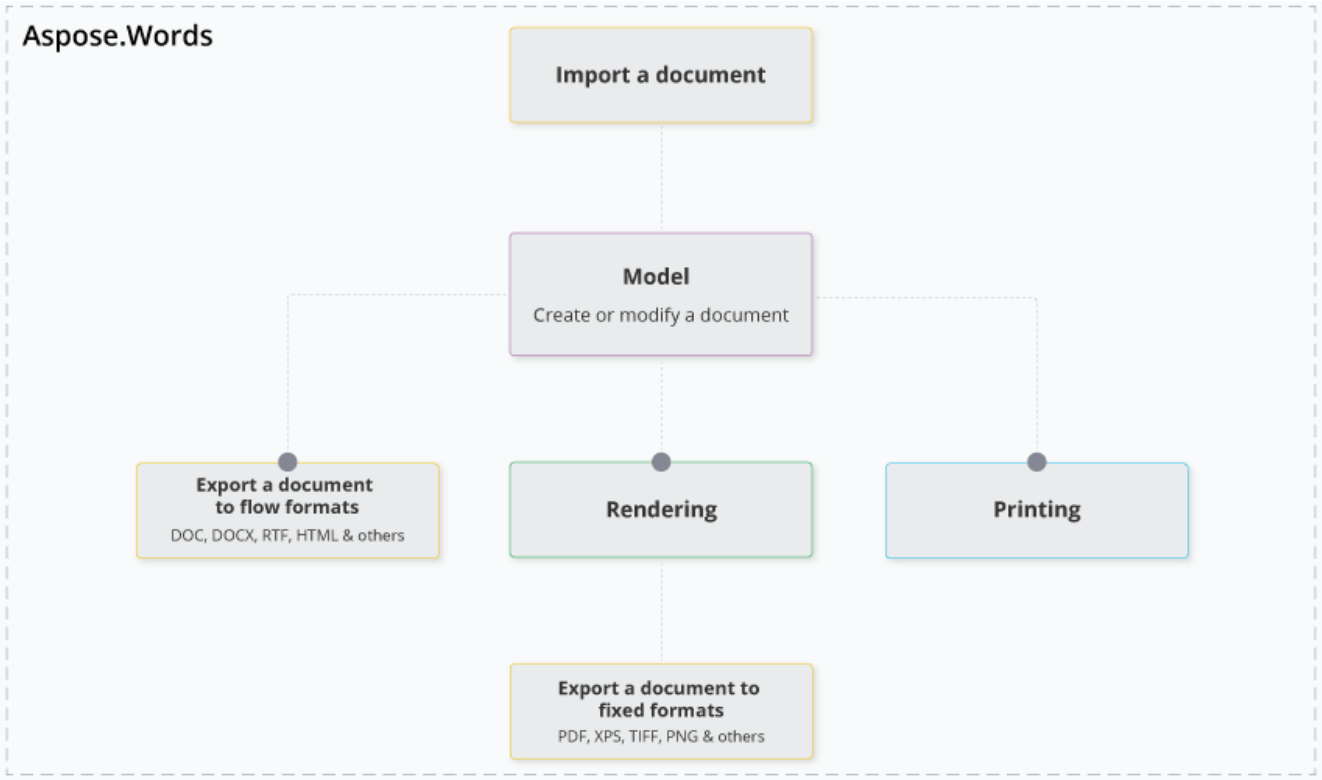


Рисунок 4.1 – Основні функції Aspose.Words для .NET

DocX – це бібліотека .NET, яка дозволяє розробникам керувати файлами Word 2007/2010/2013 у простий та інтуїтивно зрозумілий спосіб. DocX швидкий, легкий і, що найголовніше, не потребує встановлення Microsoft Word або Office. DocX – це безкоштовна версія Xceed Words для .NET з відкритим кодом [14].

Головні функції бібліотеки [14]:

– створення нових документів Word;
– змінення документів Word;
– підтримує .DOCX від Word 2007 і новіших версій;
– зміна кількох документів паралельно для кращої продуктивності;
– застосування шаблону до документа Word;
– об'єднання документів;
– підтримує захист документів паролем або без нього;
– встановлення полів документа та розмір сторінки;
– встановлення міжрядкового інтервалу, відступу, напрямку тексту, вирівнювання тексту;

– керування шрифтами та розмірами шрифтів;

– встановлення кольору тексту, налаштувань: жирний, підкреслений, курсив, закреслення, виділення;

– встановлення нумерації сторінок;

– створення розділів.

DocX робить створення документів і маніпулювання ними простим завданням. Він не використовує бібліотеки COM і не потребує встановлення Microsoft Office.

Створення файлів у форматі DOCX з використанням COM (Component Object Model) включає в себе роботу з Microsoft Word, оскільки DOCX – це формат файлів Word. Для цього можна використовувати бібліотеку Interop для взаємодії з Word з допомогою COM-технологій.

COM Interop (Взаємодія з COM) – технологія, включена до .NET CLR, що дозволяє об'єктам COM взаємодіяти з об'єктами .NET, і навпаки [15]. Завданням COM Interop є забезпечення доступу до існуючих компонентів COM без необхідності модифікації оригінальних компонентів. Ця технологія намагається зробити типи .NET еквівалентними типу COM. Крім того, COM Interop дозволяє розробникам COM отримати доступ до об'єктів, що керуються, так само просто, як і доступ до інших об'єктів COM.

Бібліотека Interop для взаємодії з Word через COM (COM Interop) в C# виглядає як обгортка навколо COM-об'єктів, що дозволяє використовувати їх у звичайному C#-коді.

Для взаємодії з Word через COM Interop в C# і створення файлів у форматі DOCX, потрібно використовувати бібліотеку Microsoft.Office.Interop.Word. Ця бібліотека надає доступ до об'єктів Word та їх функцій.

Ключові класи та методи бібліотеки [16]:

– Microsoft.Office.Interop.Word.Application. Цей клас представляє екземпляр програми Word. Використовуйте його для створення нових документів та взаємодії з існуючими;

– Microsoft.Office.Interop.Word.Document. Цей клас представляє документ Word. Використовуйте його для додавання тексту, зображень та інших елементів;

– Microsoft.Office.Interop.Word.Paragraph. Цей клас представляє абзац у документі Word. Використовуйте його для додавання тексту;

– Microsoft.Office.Interop.Document.SaveAs2. Метод SaveAs2 використовується для збереження документу;

– Microsoft.Office.Interop.Document.Close.Метод Close використовується для закриття документу;

– Microsoft.Office.Interop.Application.Quit. Метод Quit використовується для закриття екземпляра Word.

4.3 Аналіз методів збереження цілісності файлів

Цілісність файлів вказує на те, чи були файли змінені або пошкоджені. Забезпечення цілісності файлів важливо для забезпечення надійності та безпеки інформації. Це особливо важливо в областях, де необхідно забезпечити конфіденційність та невідмінність даних, таких як фінансові та медичні системи. Цілісність файлів у системі документообігу ветеринарної клініки є критичною для забезпечення правильної та безпечної роботи клініки. Деякі аспекти та заходи, які використовуються для забезпечення цілісності файлів у такій системі:

– автентифікація та авторизація. Необхідно забезпечити, щоб доступ до файлів був обмежений та контрольований за допомогою механізмів автентифікації та авторизації. Тільки уповноважені користувачі повинні мати доступ до важливих документів;

– системи контролю версій. Необхідно відстежувати зміни у документах. Це дозволить переглядати попередні версії документів та відновлювати їх у випадку потреби;

– резервне копіювання даних. Необхідно регулярно створювати резервні копії важливих даних і переконатися, що процедури відновлення даних працюють ефективно. Це дозволить відновити дані в разі втрати чи пошкодження;

– журналювання подій (event logging). Ведення журналу подій дозволяє відстежувати, хто, коли і яким чином отримував доступ до файлів. Це може бути корисним для виявлення потенційних загроз безпеки;

– хеш-суми (hash sums). Генерація унікального коду (хеша) для файлу, який можна перевірити, щоб впевнитися, що файл не був змінений. Популярні алгоритми хешування включають MD5, SHA-1 та SHA-256;

– цифровий підпис (digital signatures). Це метод, при якому файли підписуються приватним ключем, і цей підпис може бути перевірений за допомогою відкритого ключа. Електронний аналог звичайного підпису, який використовується для підтвердження автентичності повідомлення чи документа та визначення відправника.

Хеш файлу (хеш-сума файлу) – це унікальний ідентифікатор файлу, який за допомогою спеціального програмного забезпечення обчислюється комп'ютером шляхом певних математичних перетворень інформації, що міститься в ньому [17]. Функції хешування використовують різні алгоритми, такі як MD5, SHA-1, SHA-256 тощо.

MD5 (Message Digest Algorithm 5) є одним з алгоритмів хеш-функцій, який створений для створення 128-бітного (16-байтового) хеш-коду з будь-яких вхідних даних, що йому подаються. Основне призначення MD5 – швидка та ефективна генерація "відбитків" (хешів) для даних [18].

Однак MD5 став відомий своєю вразливістю до колізій. Колізія – це ситуація, коли два різних вхідних набори генерують однаковий хеш-код. Це робить MD5 непридатним для застосувань, де важливо уникнути можливості навмисної або випадкової зміни даних.

SHA-1 (Secure Hash Algorithm 1) – це алгоритм хеш-функції, який генерує 160-бітний (20-байтовий) хеш-код, що представляє собою унікальний відбиток

даних [19]. SHA-1 був розроблений для використання в криптографії та інших застосуваннях, де необхідно забезпечити високий рівень стійкості до колізій. Проте, з часом було виявлено вразливості в SHA-1, які дозволяли здійснювати атаки колізій з високою ефективністю. У зв'язку із цим, з 2020 року NIST та інші організації рекомендують уникати використання SHA-1 для криптографічних цілей. Якщо потрібно використовувати алгоритм хеш-функції, для більш високої стійкості до атак колізій рекомендується використовувати більш сучасні алгоритми, такі як SHA-256 або SHA-3.

SHA-2 (англ. Secure Hash Algorithm Version 2 – безпечний алгоритм хешування, версія 2) – сімейство криптографічних алгоритмів – однонаправлених хеш-функцій, що включає алгоритми SHA-224, SHA-256, SHA-384, SHA-512, SHA -512/256 та SHA-512/224 [20].

SHA-256 є криптографічним хеш-алгоритмом (або функцією), який використовується для перевірки цілісності повідомлень, файлів і даних. Це частина сімейства хеш-функцій SHA-2, який використовує 256-бітний ключ, щоб перетворити дані на новий, нерозпізнаний рядок даних фіксованої довжини. Цей рядок випадкових символів і чисел, який називається хеш-значенням, також має розмір 256 біт.

Три властивості роблять SHA-256 безпечним. По-перше, практично неможливо відновити вихідні дані з хеш-значення. Для генерації вихідних даних для brute-force attack знадобиться 2^{256} спроб. По-друге, наявність двох повідомлень з однаковим значенням хеш-функції (що називається зіткненням) є вкрай малоймовірною. З 2^{256} можливими хеш-значеннями (більше, ніж кількість атомів у відомому Всесвіті), ймовірність того, що два будуть однаковими, нескінченно мала, неймовірно мала. Нарешті, незначна зміна вихідних даних змінює хеш-значення настільки, що стає неочевидним, що нове хеш-значення отримано з подібних даних, це відоме як ефект лавини.

Важливо вибрати відповідний алгоритм хешування в залежності від конкретних вимог безпеки та використовувати його правильно у контексті застосування. Наприклад, у зв'язку із зростанням обчислювальної потужності, деякі

старі алгоритми, такі як MD5 та SHA-1, вважаються небезпечними для застосування в критичних системах, і рекомендується використовувати більш безпечні алгоритми, такі як SHA-256.

5 ОГЛЯД МЕТОДІВ ТА ТЕХНОЛОГІЙ ПРОЄКТУВАННЯ ТА РОЗРОБКИ ЗАСТОСУНКУ

5.1 Технології реалізації серверної частини застосунку

Серверна частина застосунку – це складова системи, яка виконує обробку запитів від клієнтської частини та забезпечує взаємодію з базою даних, іншими сервісами або зовнішніми ресурсами. Вона складається з сервера, бази даних і логіки застосунків, які працюють разом для обробки та керування даними.

Розглянемо детальний опис окремих аспектів серверної частини застосунку:

– вебсервер. Основна роль вебсервера полягає у прийомі HTTP-запитів від клієнтської сторони та надсиланні відповідей. Це може бути Apache, Nginx, чи інший вебсервер;

– маршрутизація. Маршрутизатор визначає, які обробники (controllers) повинні бути викликані для конкретних URL-шляхів. Він визначає логіку переходу від URL до конкретного обробника запиту;

– контролери (Controllers). Контролери виконують бізнес-логіку застосунку. Вони отримують дані від користувача, обробляють їх та взаємодіють із моделлю для зберігання чи витягування інформації;

– модель (Model). Модель представляє собою структуру даних та виконує операції зберігання, витягування, оновлення та видалення даних з бази даних;

– база даних. Це сховище для зберігання даних застосунку. Серверна частина взаємодіє із базою даних, виконуючи SQL-запити або використовуючи ORM (Object-Relational Mapping) для більш високорівневої роботи з даними;

– система аутентифікації та авторизації. Визначається та реалізується система для перевірки ідентифікації користувачів та надання їм відповідних прав доступу;

– API (Application Programming Interface). API визначає методи та формати обміну даними між серверною та клієнтською частинами застосунку. Це може бути RESTful API, GraphQL або інші типи API;

– логування та аудит. Реалізується система логування для відстеження дій користувачів та аудиту подій, що відбуваються в системі;

– безпека. Система безпеки включає в себе заходи для захисту від атак, таких як обробка та валідація введених даних, шифрування ідентифікаційної інформації та інші заходи безпеки.

Враховуючи усе вище сказане, визначимо основні завдання серверної частини застосунку:

– прийняття та обробка HTTP-запитів або інших типів запитів, які надходять від клієнтської частини застосунку;

– виконання необхідних операцій та обчислень для реалізації функціональності застосунку;

– зберігання та отримання даних з бази даних;

– аутентифікація та авторизація. Забезпечення доступу до ресурсів тільки авторизованим користувачам;

– застосування стратегій кешування для покращення продуктивності та зменшення навантаження на базу даних;

– взаємодія з іншими сервісами. Якщо застосунок використовує зовнішні сервіси, то серверна частина повинна взаємодіяти з ними через відповідні API;

– виявлення та обробка помилок, які можуть виникнути в процесі виконання операцій;

– захист від атак та вразливостей, включаючи валідацію введених даних, обробку винятків та інші заходи безпеки.

5.1.1 Огляд мови програмування C#

C# – об’єктно-орієнтована, компонентно-орієнтована мова програмування. C# надає мовні конструкції для безпосередньої підтримки цих концепцій, що робить C# природною мовою для створення та використання програмних компонентів. З моменту свого створення C# додав функції для підтримки нових робочих навантажень і новітніх практик проектування програмного забезпечення. За своєю суттю C# є об’єктно-орієнтованою мовою. Розробник визначає типи та їх поведінку [21].

Програми C# виконуються на .NET, віртуальній системі виконання, що називається середовищем загальномовного виконання (CLR), і наборі бібліотек класів. CLR – це реалізація Microsoft спільної мовної інфраструктури (CLI), міжнародного стандарту. CLI є основою для створення середовищ виконання та розробки, у яких мови та бібліотеки безперебійно працюють разом [21].

C# має синтаксис, подібний до C і C++, що полегшує розуміння для розробників, знайомих із цими мовами. Мова підтримує об'єктно-орієнтоване програмування, включаючи класи, об'єкти, успадкування та поліморфізм.

Введення .NET Core та .NET 5+ зробило C# платформонезалежним, що дозволяє розробляти крос-платформені застосунки для різних операційних систем. Мова має строгу систему типів, що дозволяє виявляти помилки на етапі компіляції. Автоматичний сбірник сміття допомагає управляти пам'яттю. C# підтримує асинхронне програмування за допомогою ключових слів async та await, спрощуючи роботу з асинхронним кодом. Мова має велику кількість бібліотек і фреймворків, що робить її придатною для розробки різноманітних застосунків, від вебзастосунків до мобільних застосунків.

В цілому, C# є потужною та універсальною мовою програмування, яка використовується для розробки різноманітних програмних застосунків на платформі Microsoft.

5.1.2 Огляд технології ASP Web API

ASP.NET Web API — це фреймворк, розроблений компанією Microsoft, призначений для створення вебслужб та API (інтерфейсів програмування застосунків) на платформі .NET. Він дозволяє розробляти інтерфейси для доступу до даних та послуг через HTTP [22]. Ключові характеристики та аспекти технології ASP.NET Web API:

– RESTful архітектура. ASP.NET Web API підтримує створення RESTful вебслужб, що сприяє простоті та ефективності взаємодії між клієнтами та серверами;

– підтримує HTTP-методи, такі як GET, POST, PUT, DELETE і інші. Кожен метод може бути призначений для виконання конкретної операції над ресурсами;

– маршрутизація. Маршрутизація визначає, які HTTP-запити будуть спрямовані до конкретних методів контролерів. ASP.NET Web API використовує атрибути маршрутизації для конфігурації шляхів до методів контролерів;

– серіалізація та десеріалізація. Фреймворк автоматично перетворює об'єкти .NET в формати даних, які можуть бути передані через HTTP, такі як JSON або XML, і навпаки;

– підтримка форматів відповіді. ASP.NET Web API може генерувати відповіді в різних форматах (JSON, XML, та інші), в залежності від запиту клієнта;

– фільтри та атрибути. Фреймворк дозволяє застосовувати фільтри та атрибути для маршрутів, контролерів та методів, що робить його високорівневим та гнучким;

– вбудована підтримка Dependency Injection. ASP.NET Web API має вбудовану підтримку Dependency Injection, що полегшує внедрення залежностей та покращує тестованість коду;

– система аутентифікації та авторизації. Дозволяє легко впроваджувати різні методи аутентифікації та авторизації, такі як OAuth, JWT, або власні схеми безпеки.

ASP.NET Web API використовується для розробки вебслужб та API, які можуть бути використані різними клієнтами, такими як вебзастосунки, мобільні застосунки, апаратне забезпечення та інші. Він інтегрується з іншими технологіями .NET, такими як ASP.NET MVC, щоб забезпечити повноцінний стек для розробки вебзастосунків на платформі Microsoft.

5.2 Технології доступу до даних

Технології доступу до даних включають в себе різні інструменти і підходи для взаємодії із системами управління базами даних (СУБД) та іншими джерелами даних.

Об’єктно-реляційне відображення (ORM) – це техніка, яка використовується для створення «мосту» між об’єктно-орієнтованими програмами та, у більшості випадків, реляційними базами даних. Іншими словами, можна розглядати ORM як рівень, який з’єднує об’єктно-орієнтоване програмування (ООП) з реляційними базами даних [23].

Основна мета ORM – це забезпечити абстракцію між реляційною базою даних і програмою, що дозволяє розробникам працювати з об'єктами даних, як звичайними об'єктами в програмі, не вдаючись до прямої маніпуляції SQL-запитами.

Під час взаємодії з базою даних за допомогою мов ООП доводиться виконувати різні операції, як-от створення, читання, оновлення та видалення (CRUD) даних із бази даних. За задумом використовується SQL для виконання цих операцій у реляційних базах даних. Хоча використання SQL для цієї мети не обов’язково є поганою ідеєю, ORM і інструменти ORM допомагають спростити взаємодію між реляційними базами даних і різними мовами ООП.

Основні концепції та функції ORM включають:

– об'єктно-реляційне відображення (Object-Relational Mapping). Відображення об'єктів класів програми на записи таблиць бази даних і навпаки. Це дозволяє використовувати об'єктно-орієнтований підхід до роботи з даними, що спрощує розробку;

– класи та таблиці. Класи програми відображаються на таблиці бази даних, а атрибути класів - на колонки таблиць. Це включає в себе відповідність між типами даних мови програмування та типами даних RDBMS;

– ORM дозволяє виражати та використовувати відносини між об'єктами (наприклад, зв'язки один-до-багатьох, багато-до-багатьох) за допомогою абстракцій, які вбудовані в об'єктну модель;

– мова запитів. ORM надає мову або інтерфейс для виконання запитів до бази даних, використовуючи об'єктну модель, а не роблячи прямі SQL-запити. Це забезпечує переносимість коду між різними системами управління базами даних;

– управління з'єднанням та транзакціями. ORM може автоматично управляти відкриттям та закриттям з'єднань з базою даних, а також робити транзакції більш безпечними та простими для використання.

Популярні ORM-фреймворки включають Entity Framework для платформи .NET, Hibernate для Java, SQLAlchemy для Python та Doctrine для PHP. Кожен з них має свої особливості та можливості, але загальна ідея полягає в спрощенні роботи з базами даних через використання об'єктно-орієнтованих підходів.

Entity Framework – це платформа ORM з відкритим кодом для застосунків .NET, яка підтримується Microsoft. Це дозволяє розробникам працювати з даними, використовуючи об’єкти доменних класів, не зосереджуючись на базових таблицях і стовпцях бази даних, де зберігаються ці дані. За допомогою Entity Framework розробники можуть працювати на вищому рівні абстракції, коли вони мають справу з даними, а також можуть створювати та підтримувати орієнтовані на дані програми з меншим кодом порівняно з традиційними програмами [24].

Entity Framework знаходиться між бізнес-сутностями (класами домену) і базою даних. Він зберігає дані, що зберігаються у властивостях бізнес-сутностей, а

також отримує дані з бази даних і автоматично перетворює їх на об’єкти бізнес-сутностей.

Основні компоненти та функціональність Entity Framework включають:

– DbContext.Це основний клас у EF, який представляє сесію взаємодії з базою даних. Він включає набір властивостей для кожної таблиці та методи для виконання операцій з базою даних;

– Entity. Класи, які відображають об'єкти бази даних. Кожен об'єкт цього класу представляє запис в таблиці бази даних;

– Code-First та Database-First підходи. EF підтримує два основних підходи до роботи з базою даних: Code-First, де база даних створюється на основі класів у коді, та Database-First, де класи створюються на основі існуючої бази даних;

– LINQ to Entities. Використовує Language Integrated Query (LINQ) для написання запитів до бази даних. Це дозволяє розробникам використовувати вирази LINQ для створення складних запитів та фільтрації даних;

– Lazy Loading. EF підтримує відкладене завантаження даних, що дозволяє автоматично завантажувати пов'язані об'єкти даних при потребі;

– Migrations. EF дозволяє автоматизувати процес міграцій бази даних, щоб зберігати та відстежувати зміни в схемі бази даних під час розвитку програми;

– підтримка різних баз даних. EF підтримує різні провайдери баз даних, включаючи SQL Server, MySQL, PostgreSQL та інші;

Entity Framework забезпечує абстракцію бази даних, спрощуючи роботу розробників та роблячи роботу з базами даних більш зручною та ефективною в середовищі .NET.

5.3 Технології реалізації клієнтської частини застосунку

Клієнтська частина застосунку – це та частина програми, яка виконується на стороні клієнта, тобто на пристрої чи веббраузері користувача. Вона відповідає за взаємодію з користувачем, відображення інтерфейсу, та інші клієнтські аспекти.

Frontend – це будь-яке візуальне зображення, з яким контактує користувач. Це всі частини, з якими вони безпосередньо взаємодіють, весь контент і стилі, кнопки та різні ефекти наведення курсора перед тим, як користувач натисне посилання, контактні форми з різними полями введення, вікна пошуку та спадні меню, макети, текст і кольори, зображення та відео. Однак справа не тільки в стилях. Це також те, наскільки швидко завантажується вебсайт, наскільки легко ним переміщатися та наскільки він доступний для людей з обмеженими можливостями. Важливо, наскільки він зручний і швидко реагує на різні пристрої та браузери. По суті, frontend – це всі частини вебпрограми, які створюють її зовнішній вигляд і відчуття.

Клієнтська частина застосунку виконує безліч функцій, які спрощують та полегшують взаємодію користувача з програмою. Деякі з основних функцій:

– інтерфейс користувача (UI). Вона надає користувачу зручний та інтуїтивно зрозумілий інтерфейс для взаємодії з програмою. Це включає в себе кнопки, тексти, поля введення, меню, таблиці та інші елементи для навігації та взаємодії з застосунком;

– взаємодія з сервером. Клієнтська частина виконує запити до сервера для отримання даних (GET-запити), надсилання даних на сервер (POST/PUT-запити), оновлення даних (PUT/PATCH-запити) та видалення даних (DELETE-запити);

– формування та відображення даних. Клієнтська частина отримує дані з сервера та відображає їх у вигляді, зручному для користувача, наприклад, у вигляді списків, таблиць, графіків або графічних елементів;

– обробка подій користувача. Вона реагує на події, які виникають від користувача, такі як кліки, наведення курсора, введення тексту, та виконує відповідні дії або запити до сервера;

– локальне зберігання даних. Клієнтська частина може зберігати частину даних локально, використовуючи технології, такі як локальна база даних, кешування або cookies, щоб покращити швидкість та ефективність застосунку;

– аутентифікація та безпека: Забезпечує механізми аутентифікації та авторизації користувачів, включаючи введення паролів, перевірку дозволів, управління сесіями та інші функції для забезпечення безпеки даних;

– маршрутизація та навігація. Клієнтська частина може мати механізми навігації між різними екранами або сторінками, використовуючи URL-адреси або кнопки для переміщення по застосунку;

– асинхронні операції та обробка помилок. Забезпечує можливість виконання асинхронних операцій, щоб програма не "замерзала" під час очікування на відповідь від сервера. Також обробляє та відображає повідомлення про помилки чи попередження користувачеві.

Клієнтська частина є важливою для користувача, тому вона повинна бути зручною, ефективною та забезпечувати приємний досвід взаємодії з застосунком.

5.3.1 Огляд технології HTML

HTML розшифровується як HyperText Markup Language. Це стандартна мова розмітки для створення вебсторінок. Вона дозволяє створювати та структурувати розділи, абзаци та посилання за допомогою елементів HTML (будівельних блоків вебсторінки), таких як теги та атрибути [25].

HTML має багато варіантів використання, а саме:

– веброзробка. Розробники використовують HTML-код для проектування того, як браузер відображає елементи вебсторінки, такі як текст, гіперпосилання та медіафайли;

– інтернет-навігація. Користувачі можуть легко переходити та вставляти посилання між пов’язаними сторінками та вебсайтами, оскільки HTML широко використовується для вставлення гіперпосилань;

– вебдокументація. HTML дає змогу впорядковувати та форматувати документи, подібно до Microsoft Word.

HTML використовує теги для визначення різних елементів на вебсторінці. Наприклад, <p> для абзаців, <h1> – <h6> для заголовків, для групування елементів та інші. Теги можуть мати атрибути, які надають додаткову інформацію про елемент. Наприклад, атрибут href для посилань <a> чи src для зображень <img>. HTML використовує структуру з вкладеними елементами для організації вмісту сторінки. Вона визначає загальну схему розмітки: <head> для метаданих та <body> для вмісту сторінки.

HTML використовується в поєднанні з CSS (Cascading Style Sheets) та JavaScript для створення візуально привабливих та функціональних вебсторінок та застосунків. Вона є основою для будь-якого вебзастосунку, який користувачі бачать та взаємодіють з ним через браузер.

5.3.2 Огляд технології CSS

Cascading Style Sheets (CSS) – це мова ієрархічних правил (таблиць стилів), що використовується для представлення зовнішнього вигляду документа, написаного на HTML або XML (включаючи різні мови XML, такі як SVG і XHTML). CSS описує, як елемент повинен відображатися на екрані [26].

Основним завданням CSS є розділення структури вебдокумента від його представлення, що дозволяє відокремити логіку вмісту та його візуальне відображення.

Основні принципи CSS включають:

– селектори. CSS використовує селектори для вибору елементів HTML, до яких будуть застосовані стилі. Селектори можуть бути базовими (наприклад, назва тегу чи класу) або більш складними, використовуючи комбінації різних умов;

– властивості. Кожна стилізована частина вебдокумента має ряд властивостей, які визначають її вигляд. Наприклад, властивість color визначає колір тексту, а властивість font-size визначає розмір шрифту;

– значення. Для кожної властивості визначаються конкретні значення. Наприклад, color: blue встановлює колір тексту на синій;

– каскадність. Коли різні правила стильового оформлення конфліктують, принцип каскаду дозволяє визначити, яке правило буде застосоване. Каскад може бути визначений вагою селектора, важливістю правила та порядком визначення;

– спадковість. Спадковість дозволяє елементам успадковувати стилі від їхніх батьківських елементів. Це полегшує створення єдиного дизайну для всього вебдокумента.

CSS використовується для створення привабливого та зручного відображення вебсторінок на різних пристроях та екранах. Він допомагає розробникам забезпечити естетичний вигляд вебзастосунків та сайтів. CSS працює у поєднанні з HTML та JavaScript, створюючи повноцінні та інтерактивні вебзастосунки для користувачів.

5.3.3 Огляд мови програмування JavaScript

JavaScript – це мова сценаріїв або мова програмування, яка дозволяє впроваджувати складні функції на вебсторінках [27]. Щоразу, коли вебсторінка відображає не просто статичну інформацію, на яку дивиться користувач, а своєчасні оновлення вмісту, інтерактивні карти, анімовану 2D/3D-графіку, музичні автомати з прокручуванням відео тощо, ймовірно, задіяний JavaScript.

Основні характеристики JavaScript включають:

– об'єктно-орієнтований підхід. JavaScript використовує об'єктно-орієнтовану парадигму програмування, що дозволяє організовувати код у вигляді об'єктів, які мають властивості та методи;

– динамічна типізація. Мова визначає типи даних автоматично під час виконання програми. Змінні можуть змінювати свій тип в процесі виконання;

– взаємодія з HTML/CSS. JavaScript дозволяє динамічно змінювати вміст та стилі HTML-елементів, що дозволяє створювати живі та інтерактивні вебсторінки;

– події та обробники подій. Взаємодія з користувачем здійснюється за допомогою обробки подій, таких як клік мишею, натискання клавіші та інші. JavaScript дозволяє визначати функції, які викликаються при виникненні певних подій;

– асинхронність та AJAX. JavaScript підтримує асинхронні запити до сервера за допомогою техніки AJAX, що дозволяє обмінюватися даними з сервером без перезавантаження сторінки;

– JSON (JavaScript Object Notation). JavaScript використовує JSON для обміну даними між сервером та клієнтом. JSON є легкочитаемим та легким для використання форматом обміну даними;

– багатофункціональність. JavaScript може використовуватися як на стороні клієнта (в браузері), так і на стороні сервера (за допомогою платформ, таких як Node.js).

Наведемо огляд того, як працює JavaScript на стороні клієнта [28]:

– браузер завантажує вебсторінку, коли ви її відвідує користувач;

– у процесі завантаження браузер перетворює сторінку та всі її елементи, такі як кнопки, ярлики та поля, що випадають, в структуру даних, звану об'єктною моделлю документа (DOM);

– двигун JavaScript браузера перетворює код JavaScript на байткод. Цей код є посередником між синтаксисом JavaScript та машиною;

– різні події, такі як клацання миші по кнопці, викликають виконання зв'язаного блоку JavaScript. Потім двигун інтерпретує байткод і вносить зміни до DOM;

– браузер відображає новий DOM.

JavaScript є однією з найпоширеніших мов програмування веброзробки і використовується для створення багатьох застосунків та інструментів, які ми використовуємо щоденно в Інтернеті.

5.3.4 Огляд технології React

React – це бібліотека JavaScript, відома своєю гнучкістю та ефективністю у створенні інтерактивних інтерфейсів користувача (UI) як для веб, так і для нативних програм. Завдяки компонентній архітектурі розробники можуть створювати елементи інтерфейсу користувача, такі як кнопки або рядки пошуку, які потім можна повторно використовувати в програмі. Це не тільки спрощує процес розробки, але й покращує обслуговуваність [29].

Однією з відмінних рис React є JSX (JavaScript XML). JSX виглядає схожим на HTML і дозволяє розробникам писати розмітку прямо в коді JavaScript. Ця унікальна можливість спрощує кодування, надаючи більш інтуїтивно зрозумілий спосіб структурування компонентів.

React базується на концепції компонентів, які є невеликими, самостійними блоками, які можна легко перевикористовувати та комбінувати. Вони можуть зберігати внутрішній стан (state), який може змінюватися під час взаємодії з користувачем або зміни у застосунку. Крім того, властивості (props) передаються в компоненти для передачі даних вниз по ієрархії компонентів.

React використовує віртуальний DOM для ефективного оновлення та маніпулювання вмістом сторінки. Замість безпосередньої маніпуляції реальним DOM, React вносить зміни в віртуальний DOM, а потім ефективно оновлює реальний DOM тільки зміненими частинами. Також він дозволяє створювати SPA,

де змінюються лише частини сторінки при взаємодії користувача, без перезавантаження всієї сторінки.

React використовується великою кількістю компаній та розробників для створення модерних та ефективних вебзастосунків. Більшість сучасних фронтенд-фреймворків і бібліотек мають взаємодію з React або використовують його концепції.

5.4 Технології реалізації мобільного застосунку

5.4.1 Огляд мови програмування Kotlin

Kotlin – це статично типізована мова програмування з відкритим кодом, яка підтримує як об’єктно-орієнтоване, так і функціональне програмування. Kotlin надає подібний синтаксис і концепції з інших мов, включаючи C#, Java і Scala [30]. Ця мова програмування була створена компанією JetBrains із метою поліпшення розробки програмного забезпечення для JVM (Java Virtual Machine). Вона представляє сучасну, статично типізовану мову програмування, яка може бути використана як для розробки Android-застосунків.

Kotlin робить майже все, що робить Java, але робить це краще. Він легший, чистіший і лаконічніший, ніж Java, особливо коли йдеться про написання класів даних і зворотних викликів. Єдиним очевидним аргументом на користь Java над Kotlin є те, що Java використовується ширше, і більшість прикладів документації Android написані на Java. Kotlin і Java обидва компілюються в байт-код, що означає, що їх можна використовувати в поєднанні в одному проекті [31].

Kotlin спроектована з урахуванням безпеки та зручності розробки. Вона має статичну типізацію, що дозволяє виявляти помилки на етапі компіляції. Крім того, синтаксис Kotlin чистий і лаконічний, що сприяє читабельності коду. Kotlin може значно зменшити кількість коду, необхідного для досягнення певного функціоналу порівняно з Java. Це може покращити продуктивність розробників та полегшити обслуговування коду.

Однією з ключових особливостей Kotlin є система безпеки від null, яка дозволяє уникнути багатьох помилок, пов'язаних з використанням null-значень.

Kotlin швидко набуває популярності в розробці Android-застосунків та інших областях програмування завдяки своїм перевагам, легкості використання та широкому спектру можливостей.

5.4.2 Огляд технології Retrofit

Retrofit – це бібліотека для роботи з HTTP-запитами в мові програмування Kotlin та Java. Вона використовується для спрощення реалізації з'єднань з вебсервером та обробки HTTP-запитів. Retrofit дозволяє здійснювати зв'язок з вебсервісами, взаємодію з RESTful API та отримувати та відправляти дані [32].

Основні особливості Retrofit:

– аннотації. Retrofit використовує аннотації для визначення параметрів запитів, заголовків та інших аспектів HTTP-запиту. Це спрощує структуру коду та робить його більш зрозумілим;

– серіалізація та десеріалізація. Retrofit автоматично перетворює дані між форматами JSON (або іншими) та об'єктами Kotlin/Java за допомогою бібліотеки для серіалізації, такої як Gson або Moshi;

– менеджери викликів. Retrofit використовує менеджери викликів (Call Adapters) для обробки результатів HTTP-викликів. Це може бути корисним для зручного управління відповідями та обробки помилок;

– інтеграція з OkHttp. Retrofit побудований на основі бібліотеки OkHttp, що дозволяє використовувати його переваги, такі як кешування, перехоплення запитів та інші;

– асинхронні запити. За замовчуванням Retrofit використовує асинхронний підхід для здійснення HTTP-запитів, дозволяючи уникнути блокування основного потоку виконання;

– підтримка HTTP-методів. Retrofit підтримує різні HTTP-методи, такі як GET, POST, PUT, DELETE та інші.

Ця бібліотека зазвичай використовується в розробці Android-застосунків, де потрібно взаємодіяти з віддаленим сервером через мережу Інтернет. Її легкий синтаксис та зручний інтерфейс робить роботу з мережевими запитами більш приємною та ефективною.

5.5 Опис архітектури програмної системи

Програмна система, на базі якої розроблюється інформаційна технологія документообігу матиме клієнт-серверну архітектуру. Клієнт-серверна архітектура – це розподілена модель, яка визначає взаємодію між клієнтськими та серверними компонентами у програмній системі. У такій архітектурі відбувається розділення обов'язків між клієнтською частиною, яка надає інтерфейс для взаємодії з користувачем, та серверною частиною, яка виконує обробку та зберігання даних.

Клієнти – це програмні компоненти або пристрої, які взаємодіють із користувачем і відправляють запити на сервер для отримання даних чи виконання певних операцій. Сервери – це програмні частини, які обробляють ці запити, виконують необхідні операції та повертають результати назад клієнтам.

Така архітектура дозволяє розподілити навантаження між клієнтськими та серверними компонентами, полегшуючи масштабування та забезпечуючи більшу гнучкість та підтримку.

Програмна система матиме п’ять компонентів: сервер, база даних, вебзастосунок, мобільний застосунок та інструменти для ведення документообігу. Сервер та база даних взаємодіють між собою, також сервер взаємодіє з двома клієнтами. При цьому і сервер, і клієнти можуть взаємодіяти з системою документообігу, адже електронні документи можуть формуватися як на основі даних з бази даних, та і на основі інформації, що введена у форми на стороні клієнта. Тобто для створення електронних документів відповідні дані вибираються з бази даних або з форм та вставляються у шаблонні варіанти документів. Після

цього ці електронні документи можуть бути надіслані або надані власникам тварин у форматі електронних документів для зручного використання та зберігання.

Архітектура програмної системи наведена на діаграмі розгортання, що зображена на рисунку 5.1.

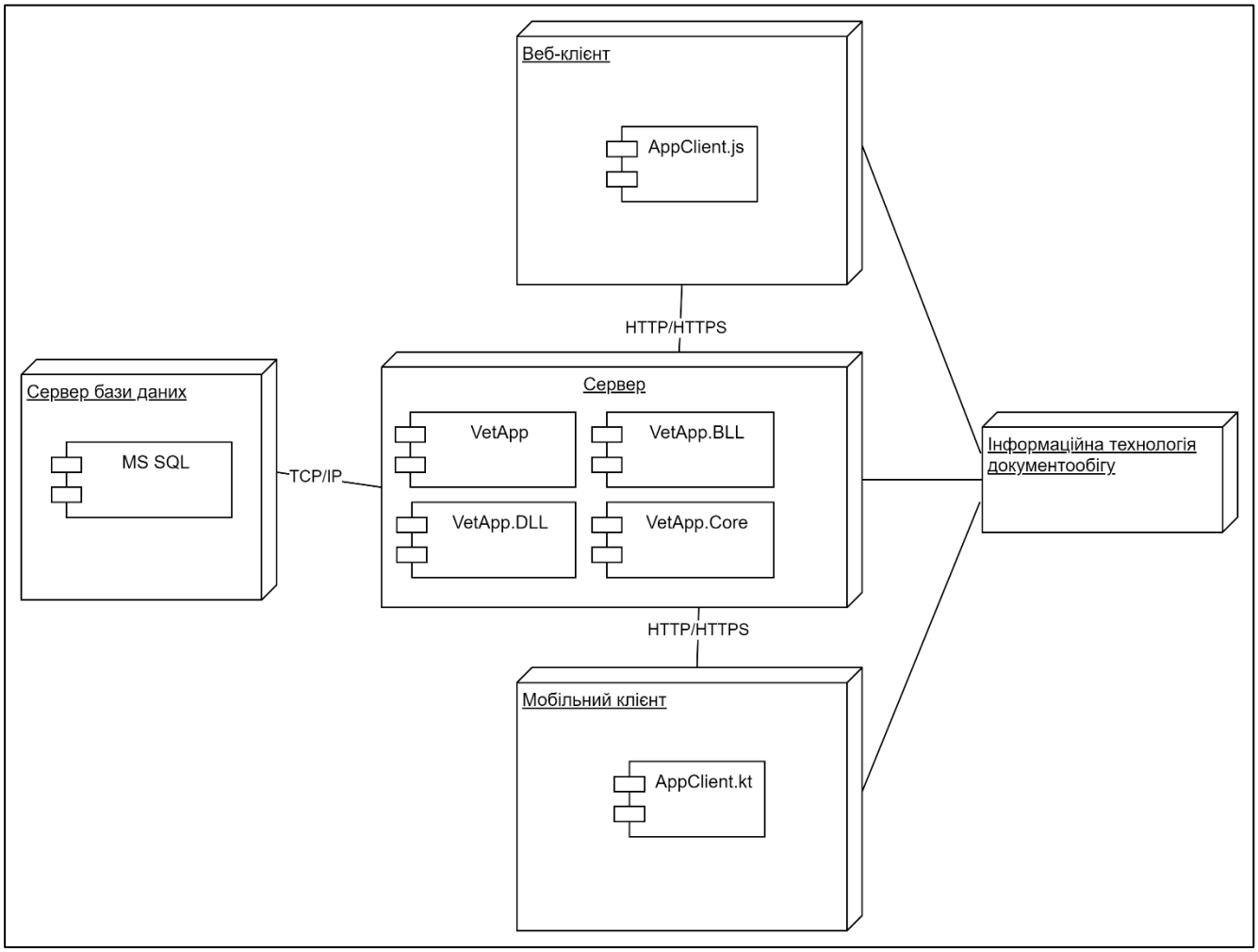


Рисунок 5.1 – Діаграма розгортання програмної системи

Серверна частина матиме багатошарову архітектуру (Layered Architecture). Компоненти в рамках шаблону багатошарової архітектури організовані в горизонтальні рівні, кожен з яких виконує певну роль у програмі [33]. Розглянемо рівні серверної частини:

– VetApp – рівень API. Це точка доступу для застосунка, у якій визначені усі кінцеві точки;
– VetApp.BLL – рівень бізнес-логіки системи;

– VetApp.DAL – рівень, що зв’язується з постачальниками даних;

Також серверна частина містить компонент VetApp.Core, що є основою системи. У ній містяться інтерфейси та моделі.

5.6 Розробка системних вимог до інформаційної технології документообігу

Для створення інформаційної технології документообігу на прикладі ветеринарної клініки буде використана операційна система Windows 10. Однак вебзастосунок буде працювати як онлайн сервіс, тому він не буде обмеженим конкретною операційною системою. Користувачі матимуть можливість використовувати вебзастосунок на комп'ютерному пристрої, який має доступ до мережі Інтернет і підтримує веббраузер. Це робить застосунок більш гнучким та доступним, оскільки він не обмежується певною операційною системою, і користувачі можуть працювати з ним на різних пристроях, таких як комп'ютери, планшети або смартфони. Такі умови не завжди дозволять користувачеві повноцінно працювати в системі. Тому наведемо рекомендовані системні вимоги, що дозволять користувачеві повноцінно працювати в вебзастосунку, й будуть гарантувати швидку реакцію системи на дії користувача:

– операційна система: Windows 10 або macOS 10.14 і вище;

– процесор: багатоядерний процесор Intel або AMD (із тактовою частотою 2 ГГц або більшою);

– оперативна пам’ять: від 4 ГБ;

– вільне місце на жорсткому диску: 1 ГБ;

– швидкість інтернет з’єднання: 100 Мбіт при належній якості з'єднання;

– розширення екрану: мінімальне розширення екрану 1280x800 пікселів;

– веббраузер: останні версії Google Chrome, Mozilla Firefox, Safari або Microsoft Edge.

Системні вимоги для використання мобільного застосунку:

- операційна система: Android версія 9.0 і вище;
- процесор: чотирьоядерний процесор ARM або еквівалент.;
- оперативна пам’ять: від 2 ГБ;
- вільне місце на жорсткому диску: 100 МБ;
- швидкість інтернет з’єднання: 100 Мбіт при належній якості з'єднання;
- розширення екрану: мінімальне розширення екрану 1280x800 пікселів.

Програмна система має складатися з трьох частин: серверної та двох клієнтів (вебзастосунок та мобільний застосунок).

Серверна частина програмної системи повинна забезпечувати взаємодію між вебклієнтом та мобільним застосунком. На цьому рівні зберігаються та структуруються дані, а також забезпечується коректна робота всіх аспектів на стороні клієнта. Для реалізації цього функціоналу використовуватиметься технологія ASP.NET Web API, а розробка буде проводитися у середовищі Visual Studio.

Клієнтська частина програмної системи повинна бути чітко відокремленою від серверної частини, забезпечуючи функціональність клієнтів у вебрежимі або у мобільному застосунку. Для реалізації вебзастосунку планується використання React та середовища розробки WebStorm. Мобільний застосунок є необхідною складовою програмної системи та має взаємодіяти з нею. Для його реалізації планується використання мови програмування Kotlin та розробка в середовищі Android Studio.

Для електронної документації буде впроваджено спеціальну папку з обмеженим доступом. Вона включає в себе каталогізацію, де кожна підпапка відповідає за унікальну збережену документацію.

Для створення інформаційної технології планується використання бази даних MS SQL Server для управління інформацією. Ця база даних буде використана для зберігання різноманітної інформації, такої як дані користувачів та інші необхідні дані, забезпечуючи надійність, ефективність та швидкий доступ до інформації.

6 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДОКУМЕНТООБІГУ

6.1 Генерація бази даних

При розробці системи була використана база даних MS SQL Server.

Був використаний Entity Framework для створення «мосту» між серверною частиною та базою даних та підхід code-first.

Entity Framework (EF) Code-First – це підход до розробки баз даних в Entity Framework, де база даних створюється автоматично з коду класів.

Для використання code-first було створено класи, які відображають таблиці бази даних. Додано атрибути та відображено відносини між таблицями.

Приклад класу:

```
using System.Collections.Generic;
using System.Collections.ObjectModel;

namespace VetApp.Core.Models
{
    public class Supplier
    {
        public Supplier()
        {
            Invoices = new Collection<Invoice>();
        }
        public int Id { get; set; }
        public string Name { get; set; }
        public string Address { get; set; }
        public string Iban { get; set; }
        public string Bank { get; set; }
        public string Mfo { get; set; }
        public string Edrpou { get; set; }
        public string Phone { get; set; }
        public string Email { get; set; }
        public string VetName { get; set; }
        public ICollection<Invoice> Invoices { get; set; }
    }
}
```

Приклад класу:

```
using System.Collections.Generic;
using System.Collections.ObjectModel;

namespace VetApp.Core.Models
{
    public class Invoice
```

```
        {
            public Invoice()
            {
                DrugsInvoice = new Collection<DrugInvoice>();
            }
            public int Id { get; set; }
            public string Date { get; set; }
            public int SupplierId { get; set; }
            public Supplier Supplier { get; set; }
            public ICollection<DrugInvoice> DrugsInvoice { get; set;
}

        }
    }
```

Створено клас, який відображає контекст бази даних та включає в себе властивості DbSet для кожного класу моделі:

```
    namespace VetApp.DAL
    {
        public class ApplicationDbContext :
IdentityDbContext<ApplicationUser>
        {
            public DbSet<Owner> Owners { get; set; }
            public DbSet<Animal> Animals { get; set; }
            public DbSet<Doctor> Doctors { get; set; }
            public DbSet<Appointment> Appointments { get; set; }
            public DbSet<Visiting> Visitings { get; set; }
            public DbSet<Direction> Directions { get; set; }
            public DbSet<Service> Services { get; set; }
            public DbSet<Schedule> Schedules { get; set; }
            public DbSet<Drug> Drugs { get; set; }
            public DbSet<Supplier> Suppliers { get; set; }
            public DbSet<Invoice> Invoices { get; set; }
            public DbSet<DrugInvoice> DrugsInvoice { get; set; }
            public
ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
: base(options)
            {

            }
            protected override void OnModelCreating(ModelBuilder
builder)
            {
                base.OnModelCreating(builder);
            }
        }
    }
```

Для з’єднання з базою даних у файл appsettings.json було додано рядок:

"ConnectionStrings": {

"ConnStr": "Data Source=DESKTOP-J5D52C8;Initial Catalog=VetApp; Integrated Security=true;"

}

Наступним кроком є включення міграції для проєкту. Приклад файлу, що містить параметри, які будуть використовуватися для міграції:

```
namespace VetApp.DAL.Configurations
{
    public class SupplierConfiguration : IEntityTypeConfiguration<Supplier>
    {
        public void Configure(EntityTypeBuilder<Supplier> builder)
        {
            builder
                .HasKey(a => a.Id);

            builder
                .Property(m => m.Id)
                .UseIdentityColumn();

            builder
                .Property(m => m.Name)
                .IsRequired()
                .HasMaxLength(50);

            builder
                .Property(m => m.Address)
                .IsRequired()
                .HasMaxLength(50);

            builder
                .Property(m => m.Iban)
                .IsRequired()
                .HasMaxLength(50);

            builder
                .Property(m => m.Bank)
                .IsRequired()
                .HasMaxLength(50);

            builder
                .Property(m => m.Mfo)
                .IsRequired()
                .HasMaxLength(50);

            builder
                .Property(m => m.Edrpou)
                .IsRequired()
```

```
                .HasMaxLength(50);

            builder
                .Property(m => m.Phone)
                .IsRequired()
                .HasMaxLength(50);

            builder
                .Property(m => m.Email)
                .IsRequired()
                .HasMaxLength(50);

            builder
                .Property(m => m.VetName)
                .IsRequired()
                .HasMaxLength(50);

            builder
                .ToTable("Suppliers");
        }
    }
}
```

Після активації міграцій було створено міграцію. Це створило скрипти для створення бази даних або внесення змін, якщо база даних вже існувала.

Застосовано міграцію для оновлення бази даних до нової схеми за допомогою команди update-database.

6.2 Генерація інформаційної технології документообігу

У результаті виконання роботи було розроблено інформаційну технологію документообігу у ветеринарній клініці.

Для сповіщення користувачів про записи на прийом були створені персоналізовані повідомлення електронною поштою. Приклад такого повідомлення наведено на рисунку 6.1.

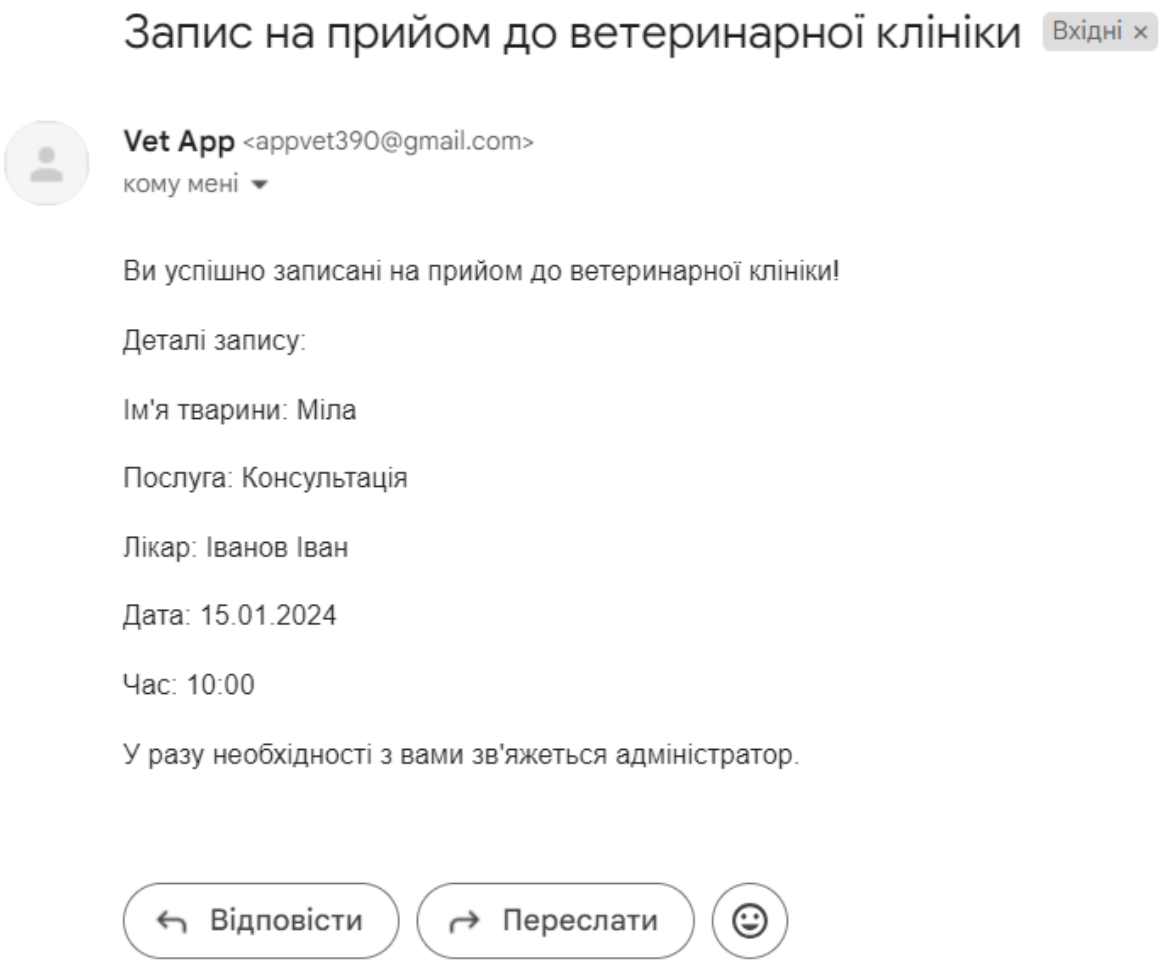


Рисунок 6.1 – Сповіщення про запис на прийом електронною поштою

На рисунку 6.2 зображено інтерфейс користувача для ведення електронної картки тварини.

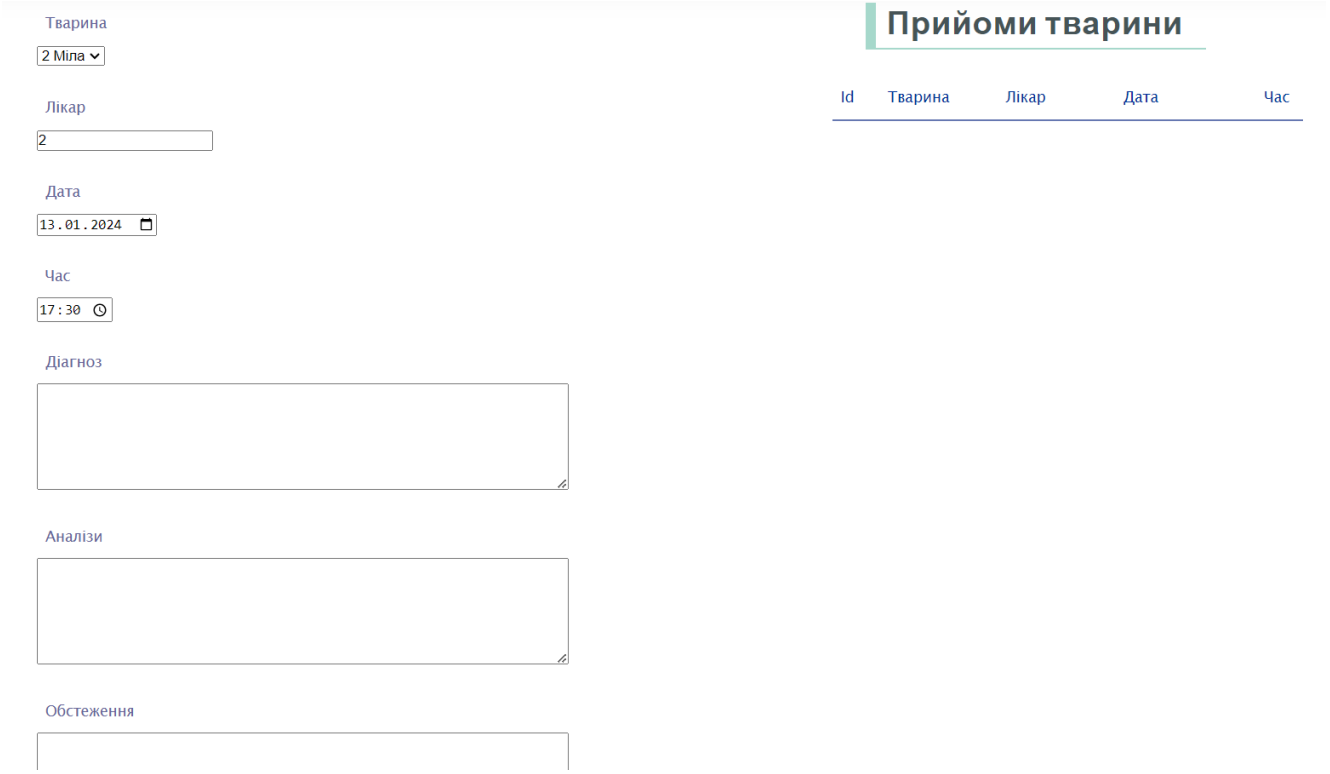


Рисунок 6.2 – Інтерфейс користувача для ведення електронної картки тварини

Для формування документів формату pdf, а саме направлень (рис. 6.3) та консультаційних висновків (рис. 6.4) було обрано бібліотеку PdfSharp. PDFsharp – легка бібліотека для створення PDF-документів у .NET. Так як створені документи мають базові елементи, і не потребують більш функціональних засобів, була обрана саме ця бібліотека.

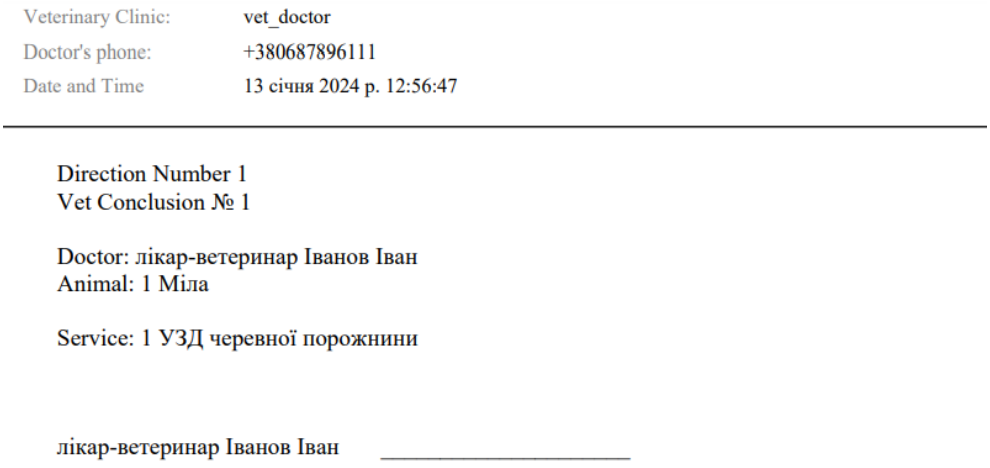
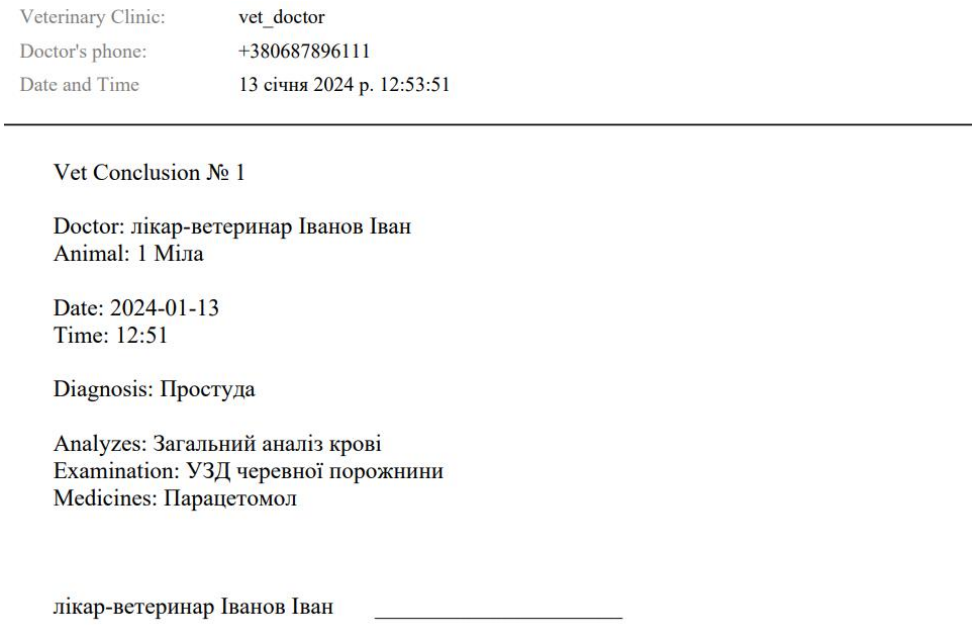


Рисунок 6.3 – Сформоване направлення у форматі pdf



Veterinary Clinic: vet_doctor
Doctor's phone: +380687896111
Date and Time 13 січня 2024 р. 12:53:51

Vet Conclusion № 1

Doctor: лікар-ветеринар Іванов Іван
Animal: 1 Міла

Date: 2024-01-13
Time: 12:51

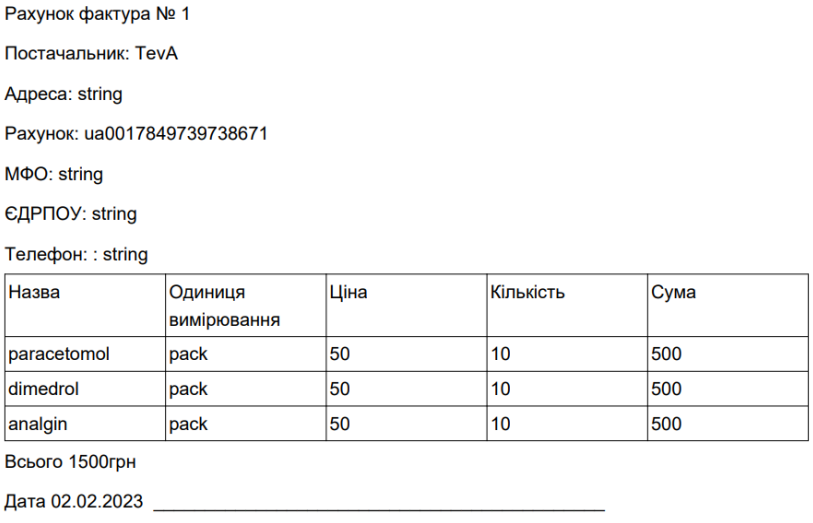
Diagnosis: Простуда

Analyzes: Загальний аналіз крові
Examination: УЗД черевної порожнини
Medicines: Парацетомол

лікар-ветеринар Іванов Іван _____

Рисунок 6.4 – Сформований висновок прийому у форматі pdf

Для створення рахунку-фактури (рис. 6.4) була обрана бібліотека iText, так як він містить таблицю. PDFsharp може бути зручним для простих завдань, але для створення таблиць може вимагає більше коду та уваги до деталей. iText має вбудований механізм для створення таблиць, що робить його зручним для складних структур даних.



Рахунок фактура № 1

Постачальник: TevA

Адреса: string

Рахунок: ua0017849739738671

МФО: string

ЄДРПОУ: string

Телефон: : string

Назва	Одиниця вимірювання	Ціна	Кількість	Сума
paracetomol	pack	50	10	500
dimedrol	pack	50	10	500
analgin	pack	50	10	500

Всього 1500грн

Дата 02.02.2023 _____

Рисунок 6.4 – Сформований рахунок-фактура у форматі pdf

ВИСНОВКИ

У результаті виконання магістерської роботи було досліджено інформаційну технологію електронного документообігу.

На прикладі ветеринарної клініки було розглянуто можливості впровадження технології електронного документообігу, яка спростить процеси роботи з документами, підвищить ефективність та забезпечить високий рівень обслуговування клієнтів.

Під час виконання роботи було проаналізовано предметну область для розуміння контексту та особливостей об'єкта дослідження, а саме описано сучасний стан розвитку інформаційних систем та технології документообігу. Також було проаналізовано застосування досліджуваної технології в існуючих системах: DocuWare, M-Files, M.E.Doc. Було визначено такі переваги використання досліжуваної інформаційної технології: збереження часу та ресурсів, зручний доступ до інформації, підвищення безпеки і конфіденційності, легка аналітика та звітність.

Для визначення основних кроків та задач автоматизації було сформульовано постановку задачі. Також було визначено, що для апробації досліджуваної інформаційної технології електронного документообігу у ветеринарних клініках необхідно розробити серверну (Back-end), клієнтську (Front-end), мобільну частини застосунку.

Було наведено математичний опис методів рішення задачі документообігу, який є важливим для того, щоб чітко визначити та формалізувати різноманітні аспекти цього процесу. Він дозволить оптимізувати потоки документів, розробляти ефективні алгоритми обробки і аналізувати ймовірність виникнення різних сценаріїв та ризиків. Також було проаналізовано методи генерації файлів та забезпечення цілісності файлів.

Також було проведено функціональне моделювання, у рамках якого були уточнені функціональні вимоги до розроблювальної інформаційної технології, визначено вхідні та вихідні дані кожної функції, визначено керуючі елементи та

нормативні дані, які регулюють та нормують виконання функції, та ресурси чи засоби, які задіяні при виконанні функції. На основі діаграм IDEF0 було побудовано сукупність діаграм потоків даних, які визначають сховища даних, потоки інформації всередині системи, місця виникнення документів, їх користувачів, і дають цілісне уявлення про документообіг в цілому у ветзакладі.

При виконанні кваліфікаційної роботи була підготовлена та опублікована наукова стаття «Дослідження інформаційної технології документообігу для оптимізації роботи ветеринарної клініки» у рамках 12-тої Міжнародної науково-технічної конференції «Інформаційні системи та технології ICT-2023» [34].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Інформаційні системи та технології : навчальний посібник. Харків: ХНАУ ім. В.В. Докучаєва, 2020. 207 с.

2. Document Workflow: The Ultimate management Guide. URL: https://www.adobe.com/acrobat/business/resources/document-workflow-ultimate-management-guide.html (дата звернення: 13.11.2023).

3. DocuWare Corporation. URL: https://www.linkedin.com/company/docuware-corporation (дата звернення: 13.11.2023).

4. M-Files. URL: https://ru.wikipedia.org/wiki/M-Files (дата звернення: 13.11.2023).

5. Getting Familiar With The User Interface. URL: https://userguide.m-files.com/user-guide/2018/eng/getting_familiar_with_the_user_interface.html (дата звернення: 13.11.2023).

6. Про M-E-Doc. URL: https://av-i.com.ua/pro-m-e-doc/ (дата звернення: 13.11.2023).

7. Електронний посібник з дисципліни: Інформаційна безпека бізнесу. https://elib.lntu.edu.ua/sites/default/files/elib_upload/%d0%9a%d0%be%d0%bd%d0%b4%d1%96%d1%83%d1%81%202%20%d0%b3%d0%be%d1%82%d0%be%d0%b2%d0%b2%d0%b0 (дата звернення: 10.01.2024).

8. What is Data Flow Diagram. https://www.visual-paradigm.com/guide/data-flow-diagram/what-is-data-flow-diagram (дата звернення: 11.01.2024).

9. Освіта в економіці підприємства. URL: https://osvita.ua/vnz/reports/econom_pidpr/21172/ (дата звернення: 13.11.2023).

10. iTextSharp in C#. URL: https://www.c-sharpcorner.com/article/itextsharp-in-C-Sharp/ (дата звернення: 13.11.2023).

11. Документація iText/Dotnet. URL: https://api.itextpdf.com/iText/dotnet/8.0.2/index.html (дата звернення: 13.11.2023).

12. Документація PDFsharp. URL: https://docs.pdfsharp.net/ (дата звернення: 14.11.2023).

13. Документація Aspose.Words для .NET. URL: https://docs.aspose.com/words/net/ (дата звернення: 13.11.2023).

14. Репозиторій DocX на GitHub. URL: https://github.com/xceedsoftware/DocX (дата звернення: 13.11.2023).

15. COM Interop In .NET. URL: https://learn.microsoft.com/en-us/dotnet/standard/native-interop/cominterop (дата звернення: 13.11.2023).

16. Документація Microsoft .NET API. URL: https://learn.microsoft.com/ru-ru/dotnet/api/ (дата звернення: 13.11.2023).

17. Що таке хеш-файла і як його визначити. URL: https://www.chaynikam.info/chto-takoe-hash-fayla-i-kak-ego-uznat.html (дата звернення: 14.11.2023).

18. MD5. URL: https://www.techtarget.com/searchsecurity/definition/MD5 (дата звернення: 14.11.2023).

19. Що таке SHA-1 і як він використовується. URL: https://tebapit.com/%D1%89%D0%BE-%D1%82%D0%B0%D0%BA%D0%B5-sha-1-%D1%96-%D1%8F%D0%BA-%D0%B2%D1%96%D0%BD-%D0%B2%D0%B8%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D0%BE%D0%B2%D1%83%D1%94%D1%82%D1%8C%D1%81%D1%8F-%D0%B4%D0%BB%D1%8F/ (дата звернення: 14.11.2023).

20. What Is Sha-2 And How Does It Work. URL: https://www.comparitech.com/blog/information-security/what-is-sha-2-how-does-it-work (дата звернення: 14.11.2023).

21. Tour Of C#. https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ (дата звернення: 02.01.2024).

22. What is Web API. https://www.tutorialsteacher.com/webapi/what-is-web-api (дата звернення: 02.01.2024).

23. What is an ORM? The meaning of object relational mapping database tools. https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of- object-relational-mapping-database-tools (дата звернення: 02.01.2024).

24. What is Entity Framework. https://www.entityframeworktutorial.net/entityframework6/what-is-entityframework .aspx (дата звернення: 02.01.2024).

25. What is HTML. https://www.hostinger.com/tutorials/what-is-html (дата звернення: 03.01.2024).

26. CSS. https://developer.mozilla.org/ru/docs/Web/CSS (дата звернення: 03.01.2024).

27. What is JavaScript. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript (дата звернення: 03.01.2024).

28. Javascript. https://aws.amazon.com/ru/what-is/javascript (дата звернення: 03.01.2024).

29. React-js. https://www.sanity.io/glossary/react-js (дата звернення: 03.01.2024).

30. Kotlin overview. https://developer.android.com/kotlin/overview (дата звернення: 03.01.2024).

31. What exactly is Kotlin. https://codeop.tech/what-exactly-is-kotlin (дата звернення: 03.01.2024).

32. Retrofit. https://square.github.io/retrofit (дата звернення: 03.01.2024).

33. Software Architecture Patterns by. https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html (дата звернення: 05.01.2024).

34. Калита Н. І., Соляник А.Р. Дослідження інформаційної технології документообігу для оптимізації роботи ветеринарної клініки.: матеріали 12-ї Міжнародної науково-технічної конференції "Інформаційні системи та технології". Частина 2. Молодіжна секція, Харків, 28 листопада 2028 – 01 грудня 2023 року /

наук. ред. В.В. Безкоровайний, L. Petryshyn, З.В. Дудар, Ю.В. Міщеряков. Харків. ХНУРЕ. 2023. с.32-34.