МІНІСТЕРСТВО ОСВІТИ І НАУКИ КРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ПУСТОВОЙТЕНКО ВЛАДИСЛАВ ВІКТОРОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій,
к.т.н., доцент
_____ О. В. Зелінська
«_____» _____ 2024 р.

# ДОСЛІДЖЕННЯ МЕТОДІВ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ОБРОБКИ ДАНИХ ПРОЦЕСІВ ОБСЛУГОВУВАННЯ КЛІЄНТІВ ВЕТЕРИНАРНИХ КЛІНІК

Спеціальність 122 «Комп'ютерні науки»

Кваліфікаційна (магістерська) робота

Науковий керівник:
Потапова Надія Анатоліївна,
доцент кафедри інформаційних технологій,
доц., к.е.н.

_____
(підпис)

Оцінка: _____/_____/_____
(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця – 2024

# АНОТАЦІЯ

**Пустовойтенко В.В.** Дослідження методів та інформаційних технологій обробки даних процесів обслуговування клієнтів ветеринарних клінік. Спеціальність 122 «Комп'ютерні науки». Освітня програма Комп'ютерна обробка даних (Data Science). Донецький національний університет імені Василя Стуса, Вінниця, 2024.

В кваліфікаційній роботі представлено розробку системи обробки інформації щодо процесів обслуговування клієнтів у ветеринарних клініках. Для створення системи прийняття рішень використовувалася мова програмування C#. Для реалізації програмного забезпечення було використано інтегроване середовище Visual Studio, за допомогою якого був розроблений прототип додатку.

Магістерська робота складається з вступу, трьох розділів, висновку та додатків. У вступі визначається актуальність теми та проводиться короткий огляд поставленої задачі. У першому розділі розглядаються теоретичні основи, методи та технології обробки даних процесів обслуговування і аналіз існуючих рішень. Другий розділ складається з створення структури бази даних, та вибір програмного забезпечення і мови розробки. Третій розділ складається з обґрунтування вибору платформи розробки, опису структури додатку та програмної реалізації. У висновках проводиться аналіз проведеної роботи та отриманих результатів.

94 с., 8 рис., 110 джерел.

Ключові слова: інформаційна технологія, аналіз даних, прийняття рішень, база даних, обслуговування клієнтів, ветеринарна клініка.

# ABSTRACT

**Pustovoitenko V.V.** Research of methods and information technologies of data processing of customer service processes of veterinary clinics. Specialization 122 "Computer Science." Educational program: Computer Data Processing (Data Science). Vasyl Stus Donetsk National University, Vinnytsia, 2024.

The qualification work presents the development of an information processing system for customer service processes in veterinary clinics. The C# programming language was predominantly used for decision-making system development. The implementation of the software utilized the Visual Studio integrated environment, which was employed to develop the application prototype.

The master's thesis consists of an introduction, three chapters, conclusions, and appendices. The introduction defines the relevance of the topic and provides a brief overview of the stated task. The first chapter covers theoretical foundations, methods and technologies of data processing of service processes and a review of existing solutions. The second chapter includes the creation of the database structure, the selection of software and programming language. The third chapter justifies the choice of the development platform, describes the application structure, and provides a software implementation. The conclusion includes an analysis of the conducted work and the obtained results.

94 pages, 8 pictures, 110 sources.

Keywords: information technology, data analysis, decision making, database, customer service, veterinary clinic.

**ЗМІСТ**

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ER-діаграма – діаграма моделі даних

ERP – система планування ресурсів підприємства

ІС – інформаційна система

C# – мова об’єктно-орієнтованого програмування

CRM – система управління взаємовідносинами з клієнтами

CDP – платформа даних клієнтів

SFA – автоматизація відділу продажів

VRM – управління взаємовідносинами з постачальниками

UX - досвід користувача (User Experience)

UI – інтерфейс користувача (User Interface)

БД – база даних (Data Base)

СУБД – система управління бази даних (System control data base)

# ВСТУП

**Актуальність теми.** У сучасному світі інформаційних технологій значуща роль приділяється ефективності обробки та управлінню даними в різних сферах діяльності. Однією з таких важливих сфер є медична галузь, а зокрема – ветеринарна медицина. Ефективність обслуговування клієнтів ветеринарних клінік є ключовим фактором задоволеності клієнтів та успішної роботи цих закладів.

Основна проблема, з якою можуть зіткнутися ветеринарні клініки, полягає у відсутності централізованого та автоматизованого підходу до обробки даних процесів обслуговування клієнтів. Наявність такого підходу дозволяє оптимізувати робочі процеси, підвищити якість обслуговування та забезпечити ефективне використання ресурсів.

**Метою роботи** є розробка та реалізація інформаційної системи для обробки та аналізу даних процесів обслуговування клієнтів, що охоплюють реєстрацію, запису на прийом, управління історією здоров'я тварин та контролю за запасами ліків у ветеринарних клініках.

**Завданнями магістерської роботи** було визначено:

1. Дослідити теоретичні положення та систематизувати основні поняття з використання інформаційних технологій та аналізу даних процесів обслуговування клієнтів ветеринарних клінік.
2. Провести аналіз існуючих додатків/веб-сайтів ветеринарних клінік, функціональне призначення яких є подібним до проектного, з метою оцінки та визначення їхні позитивних сторін та недоліків.
3. Дослідити інструментальні засоби розробки системи аналізу даних процесів обслуговування клієнтів ветеринарної клініки з метою обґрунтування елементів їх практичного використання.
4. Розробити систему для обробки даних процесів обслуговування

клієнтів ветеринарних клінік на основі авторського проекту бази даних та інтерфейсу користувача з реалізацією основних варіантів використання.

5. Спроектувати та здійснити практичну реалізацію користувацького інтерфейсу системи обслуговування клієнтів ветеринарної клініки для реалізації можливих варіантів використання.

**Об'єкт дослідження.** Процеси інформатизації обслуговування клієнтів в ветеринарних клініках.

**Предмет дослідження.** Методи та технології обробки даних процесів обслуговування клієнтів у ветеринарних клініках.

**Наукова новизна дослідження** полягає в удосконаленні процесів обслуговування клієнтів ветеринарних клінік на основі запровадження системи онлайн-реєстрації та забезпечення взаємодії клієнтів з лікарями через онлайн-повідомлення та запити.

**Структура роботи.** Структуру магістерської роботи складають: вступ, три розділи, висновки та список використаних джерел. У першому розділі систематизовано теоретичні основи систем прикладного програмного забезпечення щодо обслуговування клієнтів ветеринарних клінік. У другому розділі було розроблено проект бази даних системи обслуговування клієнтів ветеринарної клініки, розглянуто інформаційні технології та основні алгоритми обробки даних застосовані при розробці додатку. У третьому розділі були викладені результати розробки системи обробки даних обслуговування клієнтів ветеринарної клініки.

Практичне значення роботи полягає у розробці та використанні системи обслуговування клієнтів у ветеринарних клініках, запис пацієнтів та їх ведення орієнтоване на онлайн-реєстрацію та обробку даних у відповідних реєстрах. Результати даного дослідження апробовано на IV Всеукраїнській науково-практичній конференції «Комп’ютерні технології обробки даних» (м. Вінниця, 8 грудня 2023 року).

# РОЗДІЛ 1
# ТЕОРЕТИЧНІ АСПЕКТИ ІНФОРМАЦІНИХ ТЕХНОЛОГІЙ ТА АНАЛІЗУ ДАНИХ В ОБСЛУГОВУВАННІ КЛІЄНТІВ ВЕТЕРИНАРНИХ КЛІНІК

## 1.1. Сутність та основні поняття прикладного програмного забезпечення

Прикладне програмне забезпечення представляє собою конкретний тип комп'ютерної програми, спрямованої на виконання конкретних функцій. Ці функції можуть включати особисті, ділові та освітні завдання. Також воно відоме як програмне забезпечення для кінцевого користувача або для підвищення продуктивності. Кожна програма розроблена для того, щоб сприяти користувачам у виконанні певних завдань, пов'язаних із продуктивністю, ефективністю та взаємодією. Відмінно від системного програмного забезпечення, прикладне програмне забезпечення має конкретну функціональність та виконує завдання, для якого воно було розроблене. Багато програм, які ми використовуємо на смартфонах, є прикладами прикладного програмного забезпечення.

Прикладне програмне забезпечення призначене для виконання різноманітних функцій, які не обмежуються, але залежать від потреб користувача. Деякі з найпоширеніших функцій прикладного програмного забезпечення:

1. Маніпулювання даними.
2. Управління інформацією.
3. Обчислення фігур.
4. Побудова візуалів.
5. Координаційні ресурси.
6. Написання звітів.
7. Створення електронних таблиць.

8. Маніпулювання зображеннями.
9. Ведення документації.
10. Розробка сайтів.
11. Розрахунок витрат.

Ваш вибір прикладного програмного забезпечення залежить від ваших конкретних потреб. Тут наведено перелік програм або, можна сказати, категорій програмного забезпечення.

Програмне забезпечення для текстової обробки використовується для форматування, вдосконалення та редагування тексту. Воно включає функції, такі як синоніми та антоніми, і дозволяє змінювати шрифти, кольори та стиль за вашим вибором за допомогою інструменту Word Art. У цьому програмному забезпеченні також доступні опції перевірки наявності помилок в граматиці та орфографії. Microsoft Word є відмінним прикладом текстового обробника.

Програмне забезпечення для електронних таблиць головним чином служить для організації даних у вигляді таблиці та виконання різноманітних обчислень. В електронних таблицях комірки використовуються для зберігання різних типів даних, таких як час, дата, текст і числа. Користувачі можуть проводити обчислення за допомогою формул і функцій. Найкращим прикладом програмного забезпечення для роботи з електронними таблицями є Microsoft Excel.

Програмне забезпечення для створення презентацій дозволяє вам візуально виражати свої думки та ідеї, оформлюючи їх у вигляді слайдів. Ви можете зробити свої презентації цікавими та інтерактивними, використовуючи елементи, такі як відео, тексти, діаграми, графіки та зображення. Microsoft PowerPoint виступає як найпрекрасніший приклад програмного забезпечення для створення презентацій.

Програмне забезпечення, яке використовується для навчання, отримало

назву академічного програмного забезпечення, оскільки воно спеціально розроблене для полегшення процесу навчання. Це включає в себе різні типи навчального програмного забезпечення. Деякими прикладами освітнього програмного забезпечення є EDX, MindPlay і Kid Pix.

Програмне забезпечення для графіки використовується для редагування візуальних даних, зображень та анімацій. Це включає в себе різноманітне редакційне програмне забезпечення. Adobe Photoshop, Unity 3D і PaintShop є прикладами графічного програмного забезпечення.

Програмне забезпечення для моделювання – це інструмент моніторингу, що дає змогу користувачеві спостерігати за операцією, не втручаючись у її виконання. Воно стає корисним у випадках, коли робота існуючої системи є недостатньо точною, передбачуваною або потенційно небезпечною. Програмне забезпечення для моделювання широко використовується в техніці, робототехніці, авіаційних системах, прогнозу погоди, тестуванні, освіті та відеоіграх. Найкращим прикладом цього програмного забезпечення є MATLAB.

Програмне забезпечення з відкритим кодом має відкритий вихідний код і права для перевірки, зміни та покращення його ким завгодно. Більшість програм з відкритим вихідним кодом доступна безкоштовно, в той час як деякі залишаються платними на умовах, які набагато менш обмежуючі.

Програмне забезпечення із закритим вихідним кодом – це протилежність програмного забезпечення з відкритим кодом. Воно є платним, має права інтелектуальної власності і не надає доступу до вихідного коду. Зазвичай супроводжується обмеженнями та умовами використання.

Програмне забезпечення для бізнесу – це підгрупа прикладного програмного забезпечення, яке в основному використовується організацією для вирішення своїх бізнес-завдань. Це спеціально розроблене програмне

забезпечення з метою полегшення конкретних бізнес-процесів. Використання бізнес-програмного забезпечення призводить до підвищення продуктивності, ефективності, точності та забезпечує регулярні звіти для проведення бізнес-аналізу. У сучасному світі кожна нова та стрімко розвиваючася бізнес-організація активно користується програмним забезпеченням для бізнесу.

Управління взаємовідносинами з клієнтами (CRM) – це програмне забезпечення, призначене для керування взаємодією компанії з поточними та потенційними клієнтами. Воно сприяє не лише ефективному взаємодії з клієнтами, але й збору, аналізу та використанню обширних обсягів даних про клієнтів для стратегічного розвитку бізнесу.

Система планування ресурсів підприємства (ERP) – це комплексне програмне забезпечення та система, яка забезпечує управління всіма ключовими аспектами діяльності та іншими бізнес-процесами в організації. ERP може автоматизувати та оптимізувати різні сфери бізнесу, такі як закупівлі та бухгалтерський облік, управління ризиками, керування проектами та ланцюгом поставок.

Програмне забезпечення для управління проектами – це застосування, яке використовується для розробки планів проектів, контролю змін, розподілу ресурсів і багатьох інших завдань. Крім того, воно допомагає користувачам здійснювати управління бюджетом і витратами, вести документацію про хід робіт, створювати звіти про досягнення та розподіляти завдання.

Програмне забезпечення бази даних, відоме також як система управління базами даних (СУБД), створене для створення та управління базою даних організації, а також для зберігання, пошуку, обробки та вилучення важливої інформації.

Програмне забезпечення для управління бізнес-процесами, відоме також як BPM-система, є інструментом автоматизації процесів. Воно визначає, звітує

та автоматизує процеси, спрямовані на оптимізацію бізнес-функцій організації.

Програмне забезпечення для управління ресурсами - це категорія прикладного програмного забезпечення, яке сприяє плануванню ресурсів, капіталу і персоналу для ефективного завершення проектів. Воно також допомагає в управлінні кількома проектами та забезпечує реальний час розподілу ресурсів.

Програмне забезпечення для навчання є підкатегорією прикладного програмного забезпечення, створеною з освітніми цілями. Це спрощує процеси вивчення та навчання, допомагаючи засвоєнню освітнього матеріалу, концепцій і процесів.

Програмне забезпечення продуктивності призначено для оптимізації виконання завдань користувачів, забезпечуючи ефективність та своєчасність. Його варіанти включають різноманітні аспекти, такі як співпраця, управління часом, робота з базами даних і створення документів. Це програми, що сприяють підвищенню загальної продуктивності організацій та користувачів.

Програмне забезпечення, спеціально розроблене для конкретної організації чи окремого користувача, визначається як прикладне програмне забезпечення, створене відповідно до їхніх бізнес-потреб. Це надзвичайно популярний та широко використовуваний тип прикладного програмного забезпечення для бізнесу. Це програмне забезпечення може виконувати різноманітні завдання та бути адаптованим до процесів конкретної організації. В результаті ви отримуєте власне програмне забезпечення для автоматизації, інтеграції та керування всіма необхідними завданнями.

Інформаційна система (ІС) - це система, яка формально об'єднує в собі соціальні, технічні та організаційні компоненти для збору, обробки, зберігання та розповсюдження інформації. З точки зору соціальних технологій, ІС включає чотири основні компоненти: завдання, персонал, структуру (або ролі) і

технологію. Іншими словами, це інтеграція різних елементів, що використовуються для опрацювання та управління даними, які використовуються для надання інформації, розширення знань та цифрових продуктів.

Комп'ютерна інформаційна система - це система, що включає в себе співдію людей і комп'ютерів, спрямовану на обробку та інтерпретацію інформації. Термін іноді використовується для простої позначення комп'ютерної системи, на якій встановлено відповідне програмне забезпечення.

Інформаційна система — це академічний об'єкт вивчення, що охоплює конкретні довідкові дані та додаткову мережу апаратного та програмного забезпечення. Її використовують люди та організації для збору, фільтрації, обробки, створення та поширення інформації. У центрі уваги знаходяться інформаційні системи, які визначаються певними межами, користувачами, процесорами, бібліотеками для зберігання, вхідними та вихідними даними, а також вищезгаданими комунікаційними мережами [8].

У численних організаціях відділ або підрозділ, що займається інформаційними системами та обробкою даних, часто відомий під назвою "інформаційних служб". Кожна конкретна інформаційна система призначена для підтримки операцій, управління та прийняття рішень. Інформаційна система представляє собою комплекс інформаційно-комунікаційних технологій (ІКТ), які використовуються організацією, а також спосіб взаємодії людей з цією технологією для підтримки бізнес-процесів.

Деякі автори визначають чітку різницю між інформаційними системами, комп'ютерними системами та бізнес-процесами. Зазвичай, інформаційні системи включають в себе компоненти інформаційно-комунікаційних технологій (ІКТ), проте їх охоплення не обмежується виключно технологічними аспектами. Важливо відзначити, що інформаційні системи розрізняються від бізнес-

процесів. Бізнес-процеси є послідовностями дій та подій, спрямованими на досягнення конкретних бізнес-цілей, тоді як інформаційні системи допомагають контролювати та оптимізувати ефективність цих бізнес-процесів.

Альтер висвітлив переваги розуміння інформаційних систем як особливого типу робочої системи. Робоча система визначається як система, де люди або машини використовують ресурси для виконання процесів і операцій з метою виробництва конкретних продуктів або послуг для клієнтів. У свою чергу, інформаційна система – це конкретний вид робочої системи, функції якої націлені на збір, передачу, зберігання, пошук, обробку та відображення інформації.

Отже, інформаційні системи взаємодіють із системами даних та бізнес-системами. Інформаційна система представляє собою комунікаційну систему, в якій дані подаються та обробляються у вигляді соціальної пам'яті. Іншим поглядом на інформаційні системи є їх розгляд як напівформальної мови, що сприяє прийняттю людських рішень та дій.

Таким чином, інформаційна система визначається як ключовий аспект в галузі організаційної інформатики. Згідно з підходом Сільвера (1995), інтелектуальна власність, яка включає програмне забезпечення, апаратне забезпечення, дані, персонал і процедури, розглядається в рамках інформаційної системи. Чжен вносить вклад у розуміння інформаційних систем, представляючи різні системні концепції та враховуючи ключові елементи, такі як середовище, межі, ціль та взаємодія.

Комп'ютерна асоціація формулює, що "експерти інформаційних систем орієнтуються на інтеграцію інформаційно-технологічних рішень та бізнес-процесів для задоволення інформаційних потреб підприємств та інших організацій".

Інформаційні системи включають в себе різноманітні типи, такі як

системи обробки транзакцій, системи підтримки прийняття рішень, системи управління знаннями, системи управління навчанням, системи управління базами даних і офісні інформаційні системи. Застосування інформаційних технологій є важливим елементом більшості інформаційних систем, орієнтованих на автоматизацію завдань, що виходять за межі можливостей людського розуму, таких як обробка великих обсягів інформації, виконання складних обчислень та управління численними паралельними процесами.

Інформаційні технології є критичним і гнучким ресурсом для менеджерів. Багато компаній визначають посаду головного інформаційного директора (CIO), який входить до складу виконавчого комітету, разом з головним виконавчим директором (CEO), фінансовим директором (CFO), головним операційним директором (COO) та головним технологічним директором (CTO). Також можливий випадок, коли CTO виконує обов'язки CIO і навпаки.

Генеральний директор з інформаційної безпеки (CISO) спрямований на керування інформаційною безпекою. Щоб сформувати інформаційну систему, необхідно об'єднати чотири компоненти:

1. Обладнання: Термін "обладнання" вказує на машини та технічні засоби. У сучасних інформаційних системах ця категорія охоплює сам комп'ютер та всі його аксесуари. До нього входять пристрої введення та виведення, засоби зберігання та комунікації. В колишніх комп’ютерних інформаційних системах апаратне забезпечення може також включати книги та чорнило. Щодо програмного забезпечення, цей термін охоплює комп'ютерні програми та всі супутні матеріали, якщо такі існують, що їх підтримують. Комп'ютерні програми представляють собою інструкції для машини, які направляють внутрішні схеми апаратної частини системи для опрацювання даних і отримання корисної інформації. Зазвичай програми зберігаються на конкретних носіях, таких як диск чи касета. В термінах докомп'ютерних інформаційних систем "програмне

забезпечення" включало в себе не лише те, як обладнання було готове до використання (наприклад, заголовки стовпців у книзі) або інструкції з їх використання (довідник для каталогу карток).

2. Дані: Дані – це фактори, які використовуються системою для отримання корисної інформації. У сучасних інформаційних системах дані зазвичай зберігаються на диску чи стрічці у формі, доступній для читання комп'ютером, поки вони не потрібні для обробки. У попередніх комп'ютерних інформаційних системах дані зберігалися у вигляді, доступному для читання.

3. Процедура: Процедура – це визначена політика функціонування інформаційної системи управління. В широкому сенсі, це визначає, як система повинна взаємодіяти та функціонувати. "Програми для людей, програмне забезпечення для апаратних засобів" – це популярна метафора, яка ілюструє ключову роль програм у взаємодії між людьми та апаратними компонентами системи.

4. Люди: Кожна система потребує учасників, і важливо враховувати внесок людей у її функціонування. Часто недооцінюється роль особи, яка може виявити найбільший вплив на успіх чи невдачу інформаційної системи. Серед них входять не лише користувачі, а й ті, хто управляє та обслуговує комп'ютери, ті, хто відповідає за утримання даних, і ті, хто підтримує комп'ютерні мережі [25]. Зворотній зв'язок також є ключовим компонентом інформаційної системи, хоча й не обов'язковим для її функціонування.

## 1.2 Методи та технології обробки даних процесів обслуговування клієнтів ветеринарних клінік

Необхідність використання методів та технологій обробки даних процесів обслуговування клієнтів ветеринарних клінік обумовлена рядом чинників:

1. Покращення якості обслуговування. Ветеринарні клініки повинні забезпечувати високу якість обслуговування своїх клієнтів і добре доглядати за

тваринами. Використання методів та технологій обробки даних дозволяє впроваджувати системи відстеження клієнтів, їх історії та медичних записів тварин. Це допомагає покращити комунікацію між ветеринарами і власниками тварин, забезпечити швидкий доступ до необхідних даних та забезпечити більш ефективне та персоналізоване обслуговування.

2. Управління клієнтськими відносинами. Обробка даних процесів обслуговування клієнтів дозволяє ветеринарним клінікам створювати та управляти базами даних клієнтів. Це дозволяє збирати інформацію про клієнтів, їхні уподобання, історію тварин і т. д. З цими даними клініки можуть ефективно вести маркетингові кампанії, залучати нових клієнтів та підтримувати відносини з існуючими.

3. Аналіз та прогнозування. Обробка даних дозволяє ветеринарним клінікам проводити аналіз та прогнозування ефективності та результативності своєї роботи. За допомогою аналізу даних можна виявити тенденції в зміні попиту на послуги, ідентифікувати популярні процедури або послуги, а також здійснити моніторинг фінансових показників клініки. Це дозволяє ветеринарним клінікам приймати обґрунтовані рішення для поліпшення своєї діяльності та задоволення потреб клієнтів.

4. Зберігання та обробка медичних записів. Медичні записи тварин мають велике значення для ветеринарів. Використання технологій обробки даних дозволяє зберігати ці записи в цифровому форматі, що полегшує доступ до них, швидкий пошук і аналіз. Це допомагає забезпечити точність діагнозів, ефективність лікування та передачу даних між різними лікарями.

Усі ці фактори свідчать про важливість теми "Методи та технології обробки даних процесів обслуговування клієнтів ветеринарних клінік" для розвитку сучасних ветеринарних послуг та покращення взаємодії між ветеринарами і клієнтами.

Актуальність дослідження методів та технологій обробки даних процесів обслуговування клієнтів ветеринарних клінік випливає з ряду факторів:

1. Зростання популяції домашніх тварин: Востаннє декілька десятиліть спостерігається значне збільшення кількості домашніх тварин, таких як собаки та коти. Це ставить перед ветеринарними клініками виклик щодо оптимізації процесів обслуговування, ведення медичних записів та забезпечення задоволення потреб клієнтів.

2. Розвиток цифрових технологій: В останні роки спостерігається швидкий розвиток цифрових технологій, таких як хмарні обчислення, Інтернет речей (IoT), штучний інтелект (ШІ) та аналітика даних. Ці технології надають ветеринарним клінікам нові можливості для збору, аналізу та використання даних клієнтів з метою покращення якості послуг та оптимізації робочих процесів.

3. Покращення якості обслуговування: Ветеринарні клініки стрімко розвиваються у напрямку надання більш персоналізованого та ефективного обслуговування. Використання методів та технологій обробки даних дозволяє створити системи відстеження клієнтів, автоматизувати процеси запису, нагадування та спілкування з клієнтами. Це сприяє покращенню комунікації з власниками тварин, забезпеченню швидкого доступу до необхідних даних та покращенню загального досвіду клієнтів.

4. Аналіз та прогнозування: Обробка даних дозволяє ветеринарним клінікам проводити аналіз та прогнозування ефективності та результативності своєї роботи. Це допомагає виявляти тенденції в зміні попиту на послуги, ідентифікувати популярні процедури або послуги, а також здійснювати моніторинг фінансових показників клініки. Це дозволяє ветеринарним клінікам приймати обґрунтовані рішення для поліпшення своєї діяльності та задоволення потреб клієнтів.

Загалом, дослідження методів та технологій обробки даних процесів

обслуговування клієнтів ветеринарних клінік має велике значення для покращення якості ветеринарної медицини, оптимізації робочих процесів та задоволення потреб клієнтів.

За допомогою методів та технологій обробки даних процесів обслуговування клієнтів ветеринарних клінік можна впроваджувати різноманітні практичні рішення. Ось кілька прикладів:

1. Системи електронних медичних записів: Замість традиційного паперового формату, ветеринарні клініки можуть використовувати електронні медичні записи для зберігання та керування медичними даними тварин. Це дозволяє швидкий доступ до історії хвороби, результатів досліджень та лікування, спрощує спільну роботу ветеринарів та забезпечує зручну передачу даних між клініками.

2. Онлайн-системи запису та нагадувань: Ветеринарні клініки можуть використовувати онлайн-системи для запису клієнтів на прийоми, що дозволяє зменшити час очікування та полегшує планування робочого часу. Також можна надсилати автоматичні нагадування клієнтам про заплановані прийоми, вакцинації чи регулярні огляди.

3. Аналітика даних для покращення процесів: Збір і аналіз даних про клієнтів та послуги дозволяє ветеринарним клінікам виявляти тенденції та патерни. Наприклад, можна визначити найпопулярніші послуги, найефективніші методи лікування, а також ідентифікувати можливі недоліки або проблемні аспекти обслуговування клієнтів.

4. Клієнтський портал або мобільний додаток: Ветеринарні клініки можуть розробляти спеціальні онлайн-платформи або мобільні додатки для клієнтів, де вони можуть здійснювати запис на прийом, спілкуватися з лікарями, переглядати результати досліджень та отримувати рекомендації щодо догляду за своїми тваринами.

Ці приклади демонструють, як методи та технології обробки даних процесів обслуговування можуть бути застосовані для поліпшення ветеринарних послуг, забезпечення зручності для клієнтів та оптимізації робочих процесів ветеринарних клінік.

CRM або управління взаємовідносинами з клієнтами - це процес, під час якого компанія чи інша організація ефективно управляє взаємодією з клієнтами, зазвичай використовуючи аналіз великих обсягів інформації та даних.

CRM-системи збирають дані з низки різних каналів зв'язку, включаючи веб-сайт компанії, телефон (багато програмне забезпечення постачається разом із програмним телефоном), електронну пошту, живий чат, маркетингові матеріали, а віднедавна й соціальні мережі.[2] Вони дозволяють компаніям дізнатися більше про свою цільову аудиторію та про те, як найкраще задовольнити їхні потреби, таким чином утримуючи клієнтів і стимулюючи зростання продажів.[3] CRM можна використовувати з минулими, теперішніми або потенційними клієнтами. Концепції, процедури та правила, яких дотримується корпорація під час спілкування зі своїми споживачами, називаються CRM. Цей повний зв'язок охоплює прямі контакти з клієнтами, такі як продажі та операції, пов'язані з обслуговуванням, прогнозування та аналіз моделей і поведінки споживачів з точки зору компанії.[4] За даними Gartner, розмір світового ринку CRM оцінюється в 69 мільярдів доларів у 2020 році.[5, 6]

Виникнення концепції управління взаємовідносинами з клієнтами належить початку 1970-х років, коли вимір задоволення споживачів визначався через регулярні опитування або анкетування на передовій лінії. У той період компанії змушені були використовувати автономні мейнфрейм-системи для автоматизації продажів, але технологічні можливості дозволяли їм класифікувати клієнтів за різними категоріями в електронних таблицях і списків. Одним із найвідоміших попередників сучасної CRM є файл Farley. Розроблений

керівником передвиборчої кампанії Франкліна Рузвельта Джеймсом Фарлі, Файл Фарлі був повним набором записів із детальним описом політичних та особистих фактів про людей, з якими ФДР і Фарлі зустрічалися або мали зустрітися. Використовуючи його, люди, з якими зустрічався ФДР, були вражені його «пригадуванням» фактів про їхню родину та про те, чим вони займалися професійно та політично.[8]

Цю тенденцію наслідували численні компанії та незалежні розробники, які намагалися максимізувати потенційний потенціал, у тому числі Том Сібель із Siebel Systems, який розробив перший продукт CRM Siebel Customer Relationship Management у 1993 році.[9] Для того, щоб конкурувати з цими новими автономними рішеннями CRM, які швидко розвиваються, створили такі компанії, які розробляють програмне забезпечення для планування ресурсів підприємства (ERP), як Oracle, Zoho Corporation [10], [10] SAP, [11] Peoplesoft (дочірня компанія Oracle з 2005 року) [12]. ] і Navision[13] почали розширювати свої можливості продажу, дистрибуції та обслуговування клієнтів за допомогою вбудованих модулів CRM. Це включало вбудовування автоматизації відділу продажів або розширеного обслуговування клієнтів (наприклад, запити, керування діяльністю) як функції CRM у їхній ERP.

Управління взаємовідносинами з клієнтами було популяризовано в 1997 році завдяки роботі Siebel, Gartner і IBM. Між 1997 і 2000 роками провідні продукти CRM були збагачені можливостями доставки та маркетингу.[14] У 1999 році компанія Siebel представила перший мобільний CRM-додаток під назвою Siebel Sales Handheld. Ідею автономної клієнтської бази, розміщеної в хмарі, незабаром прийняли інші провідні постачальники того часу, зокрема PeopleSoft (придбана Oracle),[12] Oracle , SAP і Salesforce.com.[15]

Перша система CRM з відкритим вихідним кодом була розроблена компанією SugarCRM в 2004 році. У цей період CRM стрімко мігрувала в хмару,

в результаті чого стала доступною для індивідуальних підприємців і невеликих команд. Це збільшення доступності викликало величезну хвилю зниження цін.[14] Приблизно в 2009 році розробники почали розглядати варіанти отримання прибутку від розвитку соціальних мереж і розробили інструменти, які допоможуть компаніям стати доступними в улюблених мережах усіх користувачів. Багато стартапів того часу виграли від цієї тенденції, щоб надати виключно соціальні CRM-рішення, включаючи Base і Nutshell.[14] Того ж року компанія Gartner організувала та провела перший саміт з питань управління взаємовідносинами з клієнтами, на якому підсумувала функції, які мають запропонувати системи, щоб класифікувати їх як рішення CRM.[16] У 2013 і 2014 роках більшість популярних продуктів CRM були пов'язані з системами бізнес-аналітики та комунікаційним програмним забезпеченням для покращення корпоративного спілкування та досвіду кінцевих користувачів. Провідною тенденцією є заміна стандартизованих CRM-рішень галузевими або зробити їх достатньо адаптованими для задоволення потреб кожного бізнесу.[17] У листопаді 2016 року Forrester опублікував звіт, у якому «визначив дев'ять найбільш значущих пакетів CRM від восьми відомих постачальників».[18]

Стратегічний CRM зосереджується на розвитку бізнес-культури, орієнтованої на клієнта.[19]

Орієнтація бізнесу на клієнтоорієнтованість (у розробці та впровадженні стратегії CRM) призведе до вдосконалення CLV.[20]

Основне завдання CRM-систем – інтеграція та автоматизація продажів, маркетингу та підтримки клієнтів. Тому ці системи зазвичай мають інформаційну панель, яка дає загальне уявлення про три функції в одному поданні клієнта, окрему сторінку для кожного клієнта, який може мати компанія. Інформаційна панель може надавати інформацію про клієнта, минулі продажі, попередні маркетингові зусилля тощо, узагальнюючи всі стосунки між клієнтом і фірмою.

Операційна CRM складається з 3 основних компонентів: автоматизації відділу продажів, автоматизації маркетингу та автоматизації обслуговування.[21]

Автоматизація відділу продажів працює на всіх етапах циклу продажів, від початкового введення контактної інформації до перетворення потенційного клієнта на реального клієнта.[22] Він реалізує аналіз стимулювання збуту, автоматизує відстеження історії облікового запису клієнта для повторних продажів або майбутніх продажів і координує продажі, маркетинг, кол-центри та роздрібні точки. Це запобігає дублюванню зусиль між продавцем і клієнтом, а також автоматично відстежує всі контакти та подальші дії між обома сторонами.[22, 23]

Автоматизація маркетингу спрямована на спрощення загального маркетингового процесу, щоб зробити його ефективнішим і ефективнішим. Інструменти CRM із можливостями автоматизації маркетингу можуть автоматизувати повторювані завдання, наприклад розсилку клієнтам автоматизованих маркетингових електронних листів у певний час або публікацію маркетингової інформації в соціальних мережах. Мета автоматизації маркетингу – перетворити потенційного клієнта на повноцінного клієнта. CRM-системи сьогодні також працюють над залученням клієнтів через соціальні мережі.[24]

Автоматизація обслуговування – це частина CRM-системи, яка фокусується на технології прямого обслуговування клієнтів. Завдяки автоматизації послуг клієнти отримують підтримку через кілька каналів, таких як телефон, електронна пошта, бази знань, портали продажу квитків, поширені запитання тощо [21].

Аналітичні системи. Роль аналітичних систем CRM полягає в аналізі даних про клієнтів, зібраних із багатьох джерел, і представленні їх, щоб бізнес-менеджери могли приймати більш обґрунтовані рішення.[25] Аналітичні системи CRM використовують такі методи, як аналіз даних, кореляція та розпізнавання

шаблонів для аналізу даних клієнтів. Ця аналітика допомагає покращити обслуговування клієнтів, знаходячи невеликі проблеми, які можна вирішити, можливо, шляхом маркетингу для різних частин споживчої аудиторії по-різному.[21] Наприклад, за допомогою аналізу купівельної поведінки клієнтської бази компанія може побачити, що останнім часом ця клієнтська база не купувала багато продуктів. Після сканування цих даних компанія може подумати про те, щоб рекламувати цю підгрупу споживачів по-іншому, щоб найкраще повідомити, як саме продукти цієї компанії можуть принести користь цій групі.[26]

Третя головна мета систем CRM полягає в об'єднанні зовнішніх зацікавлених сторін, таких як постачальники, продавці та дистриб'ютори, і обмін інформацією про клієнтів між групами/відділами та організаціями. Наприклад, відгуки можна зібрати з дзвінків у службу технічної підтримки, які можуть допомогти надати напрямки щодо маркетингу продуктів і послуг цьому конкретному клієнту в майбутньому.[27]

Платформа даних клієнтів (CDP) – це комп'ютерна система, що використовується відділами маркетингу, яка збирає дані про окремих людей з різних джерел в одну базу даних, з якою можуть взаємодіяти інші програмні системи.[28] Станом на лютий 2017 року було близько двадцяти компаній, які продавали такі системи, і їх дохід становив близько 300 мільйонів доларів США [28].

Основними компонентами CRM є побудова та управління відносинами з клієнтами за допомогою маркетингу, спостереження за відносинами, коли вони розвиваються через окремі фази, управління цими відносинами на кожному етапі та визнання того, що розподіл цінності відносин для фірми не є однорідним. Розбудовуючи та керуючи відносинами з клієнтами за допомогою маркетингу, фірми можуть отримати вигоду від використання різноманітних інструментів, які допомагають організаційному дизайну, схемам стимулювання, структурам

клієнтів тощо, щоб оптимізувати охоплення своїх маркетингових кампаній. Завдяки визнанню окремих етапів CRM компанії зможуть отримати вигоду від того, що взаємодію кількох відносин розглядають як пов'язані транзакції. Останній фактор CRM підкреслює важливість CRM через облік прибутковості відносин із клієнтами. Вивчаючи особливості споживацьких звичок клієнтів, фірма може приділяти різні ресурси та кількість уваги різним типам споживачів.[29]

Реляційний інтелект, тобто усвідомлення різноманітності відносин, які клієнт може мати з фірмою, і здатність фірми зміцнювати або змінювати ці зв'язки, є важливим компонентом основних етапів CRM. Компанії можуть добре фіксувати демографічні дані, такі як стать, вік, дохід та освіта, і пов'язувати їх з інформацією про покупки, щоб класифікувати клієнтів за рівнями прибутковості, але це лише промисловий погляд фірми на відносини з клієнтами.[30] Відсутність реляційного інтелекту є ознакою того, що фірми все ще розглядають клієнтів як ресурси, які можна використовувати для можливостей продажу дорожчих або перехресних продажів, а не людей, які шукають цікавої та персоналізованої взаємодії.[31]

CRM системи включають:

1. Технологія сховища даних, яка використовується для агрегування інформації про транзакції, об'єднання інформації з продуктами CRM і надання ключових показників ефективності.

2. Управління можливостями, яке допомагає компанії керувати непередбачуваним зростанням і попитом, а також запровадити хорошу модель прогнозування для інтеграції історії продажів із прогнозами продажів.[32]

3. Системи CRM, які відстежують і вимірюють маркетингові кампанії в кількох мережах, відстежуючи аналіз клієнтів за кліками клієнтів і продажами.

4. Деяке програмне забезпечення CRM доступне як програмне

забезпечення як послуга (SaaS), яке доставляється через Інтернет і доступ до якого здійснюється через веб-браузер, а не встановлюється на локальному комп'ютері. Підприємства, які використовують програмне забезпечення, не купують його, але зазвичай платять регулярну плату за підписку постачальнику програмного забезпечення.[21]

5. Для малого бізнесу система CRM може складатися з системи керування контактами, яка об'єднує електронні листи, документи, завдання, факси та планування для окремих облікових записів. CRM-системи, доступні для конкретних ринків (юридичний, фінансовий), часто зосереджені на управлінні подіями та відстеженні відносин, а не на фінансовій окупності інвестицій (ROI).

6. Системи CRM для електронної комерції, зосереджені на задачах автоматизації маркетингу, як-от порятунок кошиків, повторне залучення користувачів електронною поштою, персоналізація.

7. Управління взаємовідносинами, орієнтоване на клієнта (CCRM), – це піддисципліна, що зароджується, і зосереджується на вподобаннях клієнтів, а не на їх впливі. CCRM прагне додати цінність, залучаючи клієнтів до індивідуальних інтерактивних стосунків.[29]

8. Системи для некомерційних організацій і організацій, заснованих на членстві, допомагають відстежувати учасників, збір коштів, демографічні показники спонсорів, рівні членства, каталоги членів, волонтерство та спілкування з окремими особами.

9. CRM не лише вказує на технологію та стратегію, але також вказує на інтегрований підхід, який включає знання співробітників, організаційну культуру для прийняття філософії CRM.

Задоволеність споживачів має важливе значення для економічних показників фірм, оскільки вона має здатність підвищувати лояльність клієнтів і поведінку користувачів, а також зменшувати скарги клієнтів і ймовірність

відходу клієнтів.[33, 34] Впровадження підходу CRM може вплинути на задоволеність клієнтів і знання клієнтів з різних причин.

По-перше, фірми можуть налаштувати свої пропозиції для кожного клієнта.[35] Накопичуючи інформацію під час взаємодії з клієнтами та обробляючи цю інформацію для виявлення прихованих закономірностей, додатки CRM допомагають фірмам налаштувати свої пропозиції відповідно до індивідуальних смаків своїх клієнтів.[35] Ця настройка покращує сприйману якість продуктів і послуг з точки зору клієнта, і оскільки сприймана якість є визначальним фактором задоволеності клієнтів, з цього випливає, що додатки CRM опосередковано впливають на задоволеність клієнтів. Додатки CRM також дозволяють компаніям забезпечувати своєчасну та точну обробку замовлень і запитів клієнтів і постійне керування обліковими записами клієнтів.[35] Наприклад, Пікколі та Епплгейт обговорюють, як Wyndham використовує IT-інструменти, щоб надавати клієнтам узгоджені умови обслуговування на різних об’єктах. Покращена здатність налаштовувати та зменшена варіативність досвіду споживання покращує сприйману якість, що, у свою чергу, позитивно впливає на задоволеність клієнтів.[36] Крім того, програми CRM також допомагають компаніям ефективніше керувати відносинами з клієнтами на етапах ініціювання, підтримки та припинення відносин.[37]

Завдяки системам управління взаємовідносинами з клієнтами клієнти краще обслуговуються в повсякденному процесі. З більш достовірною інформацією їхній попит на самообслуговування з боку компаній зменшиться. Якщо немає потреби взаємодіяти з компанією з різних проблем, рівень задоволеності клієнтів зростає.[38] Ці основні переваги CRM будуть гіпотетично пов’язані з трьома видами власного капіталу: відносини, цінність і бренд, і, зрештою, з капіталом клієнта. Було визнано вісім переваг, які забезпечують цінність.[39]

1. Покращена можливість орієнтуватися на прибуткових клієнтів.

2. Інтегрована підтримка між каналами.

3. Підвищення ефективності та результативності відділу продажів.

4. Покращене ціноутворення.

5. Індивідуальні продукти та послуги.

6. Підвищення ефективності та результативності обслуговування клієнтів.

7. Індивідуалізовані маркетингові повідомлення також називають кампаніями.

У 2012 році після перегляду попередніх досліджень хтось вибрав деякі з тих переваг, які є більш значущими для задоволеності клієнтів, і узагальнив їх у наступних випадках:[40]

Покращте обслуговування клієнтів: загалом у клієнтів виникнуть запитання, занепокоєння чи побажання. Послуги CRM надають компанії можливість створювати, розподіляти та керувати запитами або чимось зробленим клієнтами. Наприклад, програмне забезпечення колл-центру, яке допомагає зв’язати клієнта з менеджером або особою, яка найкраще може допомогти їм у вирішенні існуючої проблеми, є однією з можливостей CRM, які можна застосувати для підвищення ефективності.[41]

Покращене персоналізоване обслуговування або індивідуальне обслуговування: персоналізація обслуговування клієнтів або індивідуальне обслуговування дозволяє компаніям покращити розуміння та отримання знань про клієнтів, а також краще знати про вподобання, вимоги та вимоги своїх клієнтів.

Реагування на потреби клієнта: ситуації та потреби клієнтів можуть бути зрозумілі фірмам, які зосереджуються на потребах і вимогах клієнтів.[42]

Сегментація клієнтів: у CRM сегментація використовується для класифікації клієнтів за певною схожістю, наприклад галуззю, роботою чи

іншими характеристиками, у подібні групи.[43] Хоча ці характеристики можуть бути одним або кількома атрибутами. Це можна визначити як поділ клієнтів на основі вже відомого хорошого дискримінатора.

Удосконалення адаптації маркетингу. Сенс адаптації маркетингу полягає в тому, що фірма або організація адаптує та змінює свої послуги чи продукти на основі представлення іншого та унікального продукту чи послуги для кожного клієнта. Щоб забезпечити задоволення потреб і вимог клієнтів, організація використовує налаштування. Компанії можуть інвестувати в інформацію від клієнтів, а потім налаштовувати свої продукти чи послуги, щоб підтримувати інтереси клієнтів.

Багатоканальна інтеграція: багатоканальна інтеграція демонструє точку спільного створення цінності клієнта в CRM. З іншого боку, вміння компанії успішно виконувати багатоканальну інтеграцію значною мірою залежить від здатності організації зібрати інформацію про клієнтів з усіх каналів і об’єднати її з іншою пов’язаною інформацією.[44]

Економія часу: CRM дозволить компаніям частіше взаємодіяти з клієнтами за допомогою персоналізованих повідомлень і засобів зв’язку, які можуть бути створені швидко та своєчасно узгоджені, і, нарешті, вони зможуть краще розуміти своїх клієнтів і, отже, з нетерпінням чекати їхніх потреб.[45]

Покращте знання клієнтів: компанії можуть створювати та покращувати продукти та послуги за допомогою інформації від відстеження (наприклад, через відстеження веб-сайту) поведінки клієнтів до смаків і потреб клієнтів.[46] CRM може сприяти конкурентній перевагі в покращенні здатності фірми збирати інформацію про клієнтів, щоб налаштувати продукти та послуги відповідно до потреб клієнтів.

Дослідження показали, що збільшення утримання клієнтів на 5% збільшує прибутки клієнтів протягом усього життя в середньому на 50% у багатьох

галузях, а також до 90% у конкретних галузях, таких як страхування.[47] Компанії, які освоїли стратегії взаємодії з клієнтами, мають найуспішніші програми CRM. Наприклад, річний прибуток MBNA Europe з 1995 року зріс на 75%. Фірма інвестує значні кошти в перевірку потенційних власників карток. Після визначення відповідних клієнтів фірма утримує 97% своїх прибуткових клієнтів. Вони впроваджують CRM, продаючи потрібні продукти потрібним клієнтам. Використання карток клієнтів фірми на 52% перевищує галузеву норму, а середні витрати на транзакцію на 30% більше. Крім того, 10% власників облікових записів запитують більше інформації про продукти перехресного продажу.[47]

Amazon також досяг великого успіху завдяки своїм клієнтським пропозиціям. Фірма реалізувала персональні привітання, спільну фільтрацію тощо для клієнтів. Вони також використовували навчання CRM для співробітників, щоб побачити, як до 80% клієнтів повторюють.[47]

Профіль клієнта – це детальний опис будь-якої конкретної класифікації клієнта, створений для представлення типових користувачів продукту чи послуги. Профілювання клієнтів – це метод, що дозволяє зрозуміти ваших клієнтів з точки зору демографічних показників, поведінки та стилю життя. Він використовується для прийняття рішень, орієнтованих на клієнта, не плутаючи обсяг проекту з особистою думкою. Загальне профілювання – це збір інформації, яка підсумовує споживчі звички на даний момент і проектує їх у майбутнє, щоб їх можна було згрупувати для маркетингових і рекламних цілей.[48] Профілі клієнтів або споживачів – це суть даних, які збираються разом із основними даними (ім’я, адреса, компанія) і обробляються за допомогою методів аналізу клієнтів, по суті, типу профілювання. Трьома основними методами профілювання клієнтів є психографічний підхід, підхід типології споживача та підхід характеристик споживача. Ці методи профілювання клієнтів допомагають

вам будувати бізнес навколо того, хто є вашими клієнтами, і допомагають приймати кращі рішення, орієнтовані на клієнта.

Консультанти стверджують, що для компаній важливо створити потужні системи CRM для покращення свого реляційного інтелекту.[49] Згідно з цим аргументом, компанія повинна визнати, що люди мають багато різних типів відносин з різними брендами. Одне дослідження проаналізувало відносини між споживачами в Китаї, Німеччині, Іспанії та Сполучених Штатах з більш ніж 200 брендами в 11 галузях, включаючи авіакомпанії, автомобілі та ЗМІ. Ця інформація є цінною, оскільки забезпечує сегментацію клієнтів за демографічними ознаками, поведінкою та цінностями. Ці типи відносин можуть бути як позитивними, так і негативними. Деякі клієнти вважають себе друзями брендів, а інші – ворогами, а деякі змішані з любов'ю до бренду. Деякі стосунки є далекими, інтимними або щось середнє.[31]

Менеджери повинні розуміти різні причини типів відносин і надавати клієнтам те, що вони шукають. Компанії можуть збирати цю інформацію за допомогою опитувань, інтерв'ю тощо з поточними клієнтами. Компанії також повинні покращити реляційний інтелект своїх систем CRM. Сьогодні компанії зберігають і отримують величезні обсяги даних через електронні листи, онлайн-чати, телефонні дзвінки тощо.[50] Однак багато компаній не використовують належним чином цей великий обсяг даних. Усе це є ознаками того, які типи відносин клієнт хоче з фірмою, і тому компанії можуть розглянути можливість інвестувати більше часу та зусиль у розбудову свого реляційного інтелекту.[30] Компанії можуть використовувати технології інтелектуального аналізу даних і веб-пошук, щоб зрозуміти реляційні сигнали. Соціальні медіа, такі як сайти соціальних мереж, блоги та форуми, також можна використовувати для збору та аналізу інформації. Розуміння клієнта та фіксація цих даних дозволяє компаніям перетворювати сигнали клієнтів в інформацію та знання, які фірма може

використовувати для розуміння бажаних відносин потенційного клієнта з брендом.[51]

Багато фірм також запровадили навчальні програми, щоб навчити працівників розпізнавати та ефективно створювати міцні стосунки між клієнтом і брендом. Інші співробітники також пройшли підготовку з соціальної психології та соціальних наук, щоб допомогти зміцнити міцні відносини з клієнтами. Представники служби підтримки клієнтів повинні бути навчені цінувати стосунки з клієнтами та навчені розуміти існуючі профілі клієнтів. Навіть фінансовий і юридичний відділи повинні розуміти, як управляти та будувати відносини з клієнтами.[52]

CRM-провайдери контакт-центрів популярні серед малого та середнього бізнесу. Ці системи кодують взаємодію між компанією та клієнтами за допомогою аналітики та ключових показників ефективності, щоб надати користувачам інформацію про те, на чому зосередити свій маркетинг і обслуговування клієнтів. Це дозволяє агентам мати доступ до історії абонентів, щоб забезпечити персоналізоване спілкування з клієнтами. Намір полягає в тому, щоб максимізувати середній дохід на користувача, зменшити відтік користувачів і зменшити неактивні та непродуктивні контакти з клієнтами.[53][54][55]

Зростає популярність ідеї гейміфікації або використання елементів ігрового дизайну та принципів гри в неігровому середовищі, наприклад у середовищі обслуговування клієнтів. Гейміфікація середовищ обслуговування клієнтів включає надання таких елементів, які можна знайти в іграх, як нагороди та бонусні бали представникам служби підтримки клієнтів, як метод зворотного зв’язку за добре виконану роботу.[56] Інструменти гейміфікації можуть мотивувати агентів, враховуючи їх бажання винагород, визнання, досягнень і конкуренції.[57]

Автоматизація контакт-центру, CCA, практика наявності інтегрованої

системи, яка координує контакти між організацією та громадськістю, призначена для скорочення повторюваних і виснажливих частин роботи агента контакт-центру. Автоматизація запобігає цьому завдяки наявності попередньо записаних звукових повідомлень, які допомагають клієнтам вирішити їхні проблеми. Наприклад, автоматизований контакт-центр може перенаправляти клієнта за допомогою серії команд, які просять його або її вибрати певний номер для розмови з конкретним агентом контакт-центру, який спеціалізується в галузі, у якій клієнт має запитання. [58] Програмні засоби також можна інтегрувати з настільними інструментами агенту для обробки запитань і запитів клієнтів. Це також економить час від імені працівників.[24]

Соціальний CRM передбачає використання соціальних мереж і технологій для залучення споживачів і навчання від них.[59] Оскільки громадськість, особливо молодь, все частіше використовує сайти соціальних мереж, компанії використовують [31] ці сайти, щоб привернути увагу до своїх продуктів, послуг і брендів, з метою налагодження відносин із клієнтами для збільшення попиту. Зі збільшенням використання платформ соціальних медіа інтеграція CRM за допомогою соціальних медіа потенційно може бути швидшим і дешевшим процесом.[60]

Деякі системи CRM інтегрують сайти соціальних мереж, як-от Twitter, LinkedIn і Facebook, для відстеження та спілкування з клієнтами. Ці клієнти також діляться власними думками та досвідом щодо продуктів і послуг компанії, даючи цим фірмам більше розуміння. Таким чином, ці фірми можуть як ділитися власними думками, так і відстежувати думки своїх клієнтів.[27]

Програмні платформи управління зворотним зв’язком підприємства поєднують дані внутрішнього опитування з тенденціями, виявленими через соціальні медіа, щоб дозволити підприємствам приймати точніші рішення про те, які продукти постачати.[61]

Системи CRM також можуть включати технології, які створюють географічні маркетингові кампанії. Системи отримують інформацію на основі фізичного місцезнаходження клієнта та іноді інтегрують її з популярними програмами GPS на основі визначення місцезнаходження. Його можна використовувати для мереж або керування контактами, а також для збільшення продажів залежно від місця розташування.[24]

Незважаючи на загальне уявлення про те, що CRM-системи були створені для клієнтів, орієнтованих на бізнес, їх також можна застосовувати в середовищі B2B для оптимізації та покращення умов керування клієнтами. Для найкращого рівня функціонування CRM у середовищі B2B програмне забезпечення має бути персоналізованим і поставлятися на індивідуальних рівнях.[62]

Основні відмінності між системами CRM «бізнес-споживач» (B2C) і «бізнес-бізнес» стосуються таких аспектів, як розмір бази даних контактів і тривалість відносин.[63]

На Gartner CRM Summit 2010 обговорювалися такі проблеми, як «система намагається отримати дані з трафіку соціальних мереж, як-от Twitter, обробляє адреси сторінок Facebook або інших онлайн-сайтів соціальних мереж», і було запропоновано рішення, які допоможуть залучити більше клієнтів.[64]

Ера «соціального клієнта» стосується використання клієнтами соціальних медіа[65].

Деякі системи CRM оснащені мобільними можливостями, що робить інформацію доступною для віддаленого торгового персоналу.[66, 67, 68]

Багато постачальників CRM пропонують веб-інструменти на основі передплати (хмарні обчислення) і SaaS. Salesforce.com була першою компанією, яка надала корпоративні програми через веб-браузер, і зберегла свої лідерські позиції.[69]

Традиційні провайдери перейшли на ринок хмарних технологій через

придбання менших провайдерів: Oracle придбала RightNow у жовтні 2011 року[70], а Taleo[71] і Eloqua[72] у 2012 році; SAP придбала SuccessFactors у грудні 2011 року[73], а NetSuite придбала Verenia у 2022 році[74].

Служби продажів також відіграють важливу роль у CRM, оскільки максимізація ефективності продажів і підвищення продуктивності продажів є рушійною силою впровадження програмного забезпечення CRM. Деякі з головних тенденцій CRM, виявлених у 2021 році, включають зосередження на автоматизації обслуговування клієнтів, наприклад чат-боти, гіперперсоналізація на основі даних і розуміння клієнтів, а також використання уніфікованих систем CRM.[75, 76] Постачальники CRM підтримують продуктивність продажів за допомогою різних продуктів, таких як інструменти, які вимірюють ефективність реклами, що з’являється в 3D-відеоіграх.[77]

Фармацевтичні компанії були одними з перших інвесторів в автоматизацію відділу продажів (SFA), а деякі впроваджують технологію третього або четвертого покоління. Однак донедавна розгортання не виходило за межі SFA, що обмежувало їх масштаб і інтерес для аналітиків Gartner.[78]

Іншою пов’язаною розробкою є управління взаємовідносинами з постачальниками (VRM), яке надає інструменти та послуги, які дозволяють клієнтам керувати своїми індивідуальними відносинами з постачальниками. Розробка VRM виросла завдяки зусиллям ProjectVRM у Гарвардському центрі Беркмана для Інтернету та суспільства та Identity Commons' Internet Identity Workshops, а також завдяки зростанню кількості стартапів і відомих компаній. VRM була темою обкладинки в журналі CRM за травень 2010 року.[79]

Ще однією тенденцією, яку варто відзначити, є зростання успіху клієнтів як дисципліни в компаніях. Все більше і більше компаній створюють групи успіху клієнтів окремо від традиційної команди продажів і доручають їм керувати існуючими відносинами з клієнтами. Ця тенденція підживлює попит на

додаткові можливості для більш цілісного розуміння здоров'я клієнтів, що є обмеженням для багатьох існуючих постачальників у просторі.[80] Як наслідок, на ринок з'являється все більше нових учасників, тоді як існуючі постачальники додають можливості в цій галузі до своїх пакетів.

### 1.3. Онлайн-клініки ветеринарної медицини та особливості їх роботи

Ветеринарна клініка – лікувально-профілактичний заклад для надання допомоги хворим тваринам на прийомі в спеціалізованому закладі. Вона має штат спеціалістів, які займаються прийомом пацієнтів за записом. Оплата прийому здійснюється після отримання амбулаторної картки, що оформлюється в реєстратурі. Сучасна ветеринарна клініка є спеціалізованою лікувально-профілактичною установою, призначеною здійснювати комплекс профілактичних заходів для виявлення хвороб у тварин та надання їм медичної допомоги.

До функцій ветеринарної клініки входить:

1. Надання першої медичної допомоги;
2. Проведення лабораторних аналізів;
3. Передчасне виявлення хвороб тварини.

Клініка проводить велику профілактичну роботу та протиепідемічні заходи, вивчає здоров'я тварини, виявляє ранню захворюваність та організовую статистичний облік.

Впровадження інформаційної системи підтримки надання медичної допомоги у діяльність медичних установ призведе до забезпечення висококваліфікованої, зручної, сучасної та швидкої медичної допомоги тваринам.

Пацієнт – фізична особа, яка отримує діагностичну, профілактичну, медичну допомогу, або піддається медико-біологічним випробуванням

(дослідженням).

Ветеринарна хірургія – наука, яка вивчає способи та правила виконання хірургічних операцій на тваринах.

Використовуючи методи та способи втручань, вона вирішує конкретні завдання:

1. Відновлення зниженої або втраченої продуктивності тварини;
2. Відновлення або поліпшення робочих якостей тварини;
3. Сприяння якнайшвидшому відтворенню стада.

Хірурги виконують такі хірургічні процедури, як:

1. Ендопротезування суглобів;
2. Відновлення переломів;
3. Стабілізація дефектів краніальних хрестоподібних зв’язків;
4. Онкологічні операції;
5. Лікування міжхребцевих трансплантів;
6. Малоінвазивні процедури та лікування ран.

У ветеринарній хірургії застосовують неоперабельні методи лікування тварин з хірургічною патологією, використовуючи обладнання для реабілітації та фізіотерапії, що дозволяє прискорити одужання тварин.

Стерильність у ветеринарії дуже важлива. Стерильність означає повну відсутність будь-яких форм бактерій, що необхідно для проведення операцій та хірургічних втручань. Стерильним повинно бути все, для попередження мікробного зараження ран. Усі інструменти стерилізуються у сухожаровій шафі при високих температурах або в автоклаві парою під тиском.

Впровадження програмного продукту для ветеринарної лікарні – це важливий етап, що вимагає дбайливого планування та врахування ряду аспектів. Ключові аспекти, які слід враховувати при цьому процесі:

1. Аналіз потреб: Почніть з ретельного аналізу потреб вашої ветеринарної

лікарні. Визначте, які конкретні завдання ви хочете вирішити за допомогою програмного продукту, чи це буде ведення медичних записів, планування прийому, облік лікарських препаратів тощо.

2. Визначення функціоналу: Розробіть перелік основних функцій, які повинен виконувати програмний продукт. Врахуйте специфічні вимоги ветеринарної галузі, такі як ведення медичних карток для тварин, реєстрація лікарських препаратів, аналіз даних про пацієнтів і так далі.

3. Системні вимоги: З'ясуйте, які системні вимоги потрібні для використання програмного продукту. Це може включати аспекти, такі як операційна система, обсяг пам'яті, швидкість процесора тощо.

4. Інтеграція з іншими системами: Якщо у вашій лікарні вже використовуються інші системи, такі як облік клієнтів чи бухгалтерія, важливо забезпечити сумісність та можливість інтеграції нового програмного продукту з існуючими системами.

5. Безпека даних: У ветеринарії важливо забезпечити конфіденційність та безпеку медичних даних. Врахуйте вимоги стосовно захисту даних пацієнтів та дотримання стандартів безпеки.

6. Тестування: Плануйте етапи тестування програмного продукту перед його повним впровадженням. Важливо переконатися, що програма працює стабільно, а її функції відповідають вимогам.

7. Тренінг персоналу: Забезпечте належний тренінг персоналу. Введіть їх в особливості використання програмного продукту, інструкції з ефективного ведення медичних записів та інші аспекти.

8. Підтримка та оновлення: Розробіть стратегію підтримки користувачів та регулярних оновлень програмного продукту для виправлення помилок та впровадження нових функцій.

9. Вартість впровадження: Розрахуйте витрати на впровадження

програмного продукту, включаючи вартість ліцензій, тренінгу персоналу, налаштування та технічну підтримку.

10. Залучення стейкхолдерів: Врахуйте думку та вимоги ключових стейкхолдерів, таких як ветеринари, адміністратори лікарні, а також побажання клієнтів.

Ці кроки допоможуть створити чіткий план впровадження програмного продукту для вашої ветеринарної лікарні, що максимально враховує ваші потреби та забезпечує ефективність використання нового рішення.

На сьогоднішній день в нашій країні існує значна кількість ветеринарних клінік, що пропонують різноманітні послуги. Більшість з них обладнані власними веб-сайтами та мобільними додатками, за допомогою яких можна здійснити онлайн-запис на прийом. До числа таких клінік входять "Пес і Кіт" та "MyPet", які вирізняються своєю доступністю та зручністю для клієнтів.

Першим розглянемо сайт «MyPet». Перевагою даного веб-сайту є його простота та зручність. Ветеринарна клініка пропонує розмаїття послуг, серед яких важко не відзначити кілька неординарних, таких як готель для тварин та "швидка допомога". Однак слабкістю є відсутність інформації про різноманітні хвороби, що могло б бути надзвичайно корисним, особливо в легких випадках захворювань. Домашню сторінку веб-сайту можна розглянути на рис 1.1 .

Наступною на черзі у нас буде розглянуто ветклініку «Пес і Кіт». На даному веб-сайті ви зможете знайти розгорнутий перелік послуг, вартість, відомості про ветеринарну клініку, останні новини та акції, а також контактні дані.

Однією із вагомих переваг цього сервісу є можливість отримати безкоштовну консультацію від фахівця у будь-який зручний момент. Для запису на прийом до «Пес і Кіт» просто залиште свій номер телефону, і представник клініки вам зателефонує.

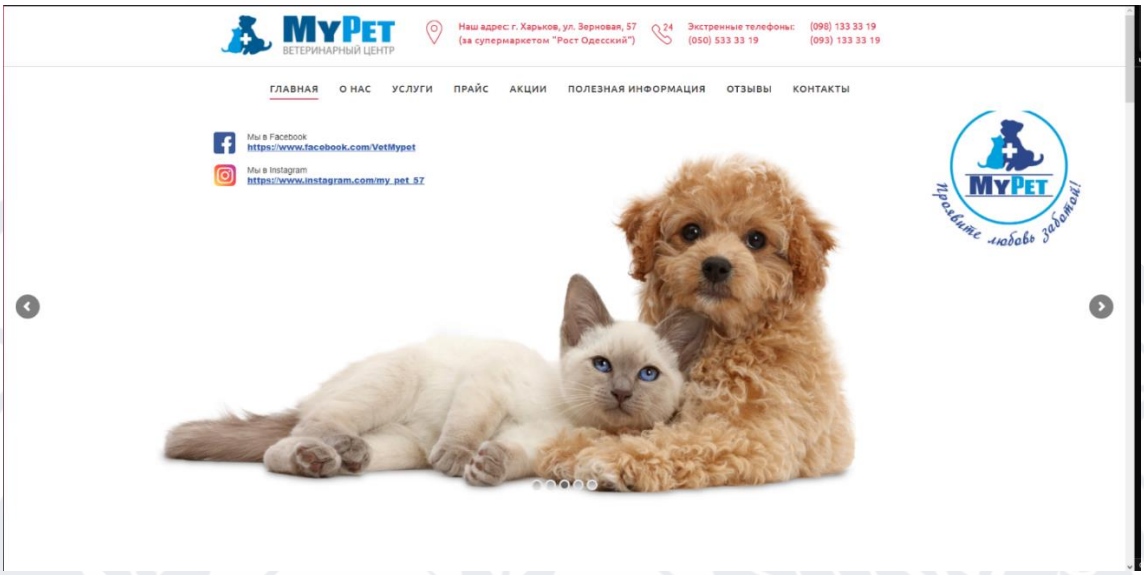


Рис.1.1 Головна сторінка MyPet

Але на жаль, є певні недоліки. Сервіс не надає можливості реєстрації для клієнтів, що може виявитися не зручним, особливо коли користувач хоче переглядати інформацію про минулі відвідування, а також відсутність української мови.

Ще одним аналогом є сайт ветеринарної клініки «Пес і Кіт». Порівнюючи аналогічні ветеринарні веб-сайти, можна сформулювати декілька висновків:

1. Зручність інтерфейсу: Обидва сайти виявляються зручними та легкими у використанні для користувачів.

2. Доступність інформації: Обидва сервіси представляють широкий перелік інформації, такої як послуги, ціни, опис клінік, новини та акції.

3. Можливості запису на прийом: "Пес і Кіт" пропонує зручний механізм запису на прийом через вказання номеру телефону, що спрощує взаємодію з клієнтами. З іншого боку, "MyPet" не надає конкретної інформації щодо процесу запису.

4. Безкоштовна консультація: Великою перевагою "Пес і Кіт" є можливість

отримати безкоштовну консультацію у будь-який час, що може бути привабливим для клієнтів.

5. Реєстрація користувачів: "MyPet" не надає можливості реєстрації для клієнтів, що може виявитися не дуже зручним, особливо для тих, хто хоче відслідковувати свої попередні візити.

6. Недоліки: Однак, в обох випадках є помітні недоліки, тобто відсутність інформації про існуючі хвороби на сайті може вплинути на інформативність.

Загальною тенденцією є те, що обидва ветеринарні сайти стараються забезпечити користувачам широкий функціонал та зручність використання, однак існують певні різниці у деталях, таких як можливості запису та реєстрації користувачів.

## ВИСНОВКИ ДО РОЗДІЛУ 1

У контексті швидко розвиваючихся технологій ветеринарна сфера може зазнати істотних перетворень завдяки використанню інформаційних технологій. Зокрема, автоматизовані інформаційні системи можуть полегшити робочий процес ветеринарних клінік, забезпечуючи швидкий та зручний доступ до інформації для клієнтів та персоналу.

Аналіз існуючих ветеринарних клінік дозволяє виявити найкращі практики та визначити слабкі сторони, що може служити підґрунтям для вдосконалення та впровадження нової інформаційної системи. Переваги такої системи можуть включати автоматизований процес запису на прийом, ефективне ведення медичних історій, а також забезпечення зручного зв'язку між клієнтами і клінікою.

Основна мета створення інформаційної системи – забезпечити покращений сервіс для клієнтів і оптимізувати внутрішні процеси клініки. Це може включати в себе створення онлайн-сервісів для запису на прийом, розробку зручних інтерфейсів для взаємодії з історіями захворювань тварин, інтеграцію засобів аналізу даних для поліпшення роботи клініки та підвищення задоволеності клієнтів.

# РОЗДІЛ 2
# ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РОЗРОБКИ СИСТЕМИ АНАЛІЗУ ДАНИХ ПРОЦЕСІВ ОБСЛУГОВУВАННЯ КЛІЄНТІВ ВЕТЕРИНАРНОЇ КЛІНІКИ

## 2.1 Проєкт бази даних системи обслуговування клієнтів

База даних, відома також як сховище даних, представляє собою систематизований набір інформації, структурований відповідно до певної концепції, що описує характеристики цих даних та їх взаємозв'язки. Згідно із стандартом ISO/IEC, база даних підтримує щонайменше одну область застосування. База даних зазвичай містить в собі структури, таблиці, види, збережені процедури та інші компоненти, а дані в ній організовані відповідно до певної моделі структури даних.

У загальному розумінні, базою даних може вважатися будь-який систематизований набір даних, такий як паперова картотека з інформацією про працівників у відділі кадрів. Однак дана стаття акцентує увагу на використанні баз даних в інформаційних системах. У сучасному світі програми для роботи з базами даних вважаються одними з найпоширеніших прикладних застосунків.

Для початку побудови бази даних спершу потрібно спроектувати і створити ER-діаграму.

ER-діаграма (Entity-relationship model або entity-relationship diagram) – це модель даних, яка дозволяє описувати концептуальні схеми, використовуючи абстрактні структури блоків.

За концептуальною моделлю предметної області можливо визначити наступні сутності:

1. Лікар;
2. Користувач;
3. Тварина;

4. Прийом;
5. Ліки;
6. Діагноз;
7. Послуги;

Зв’язок та атрибути вказаних сутностей наведено на ER-діаграмі (рис 2.1)

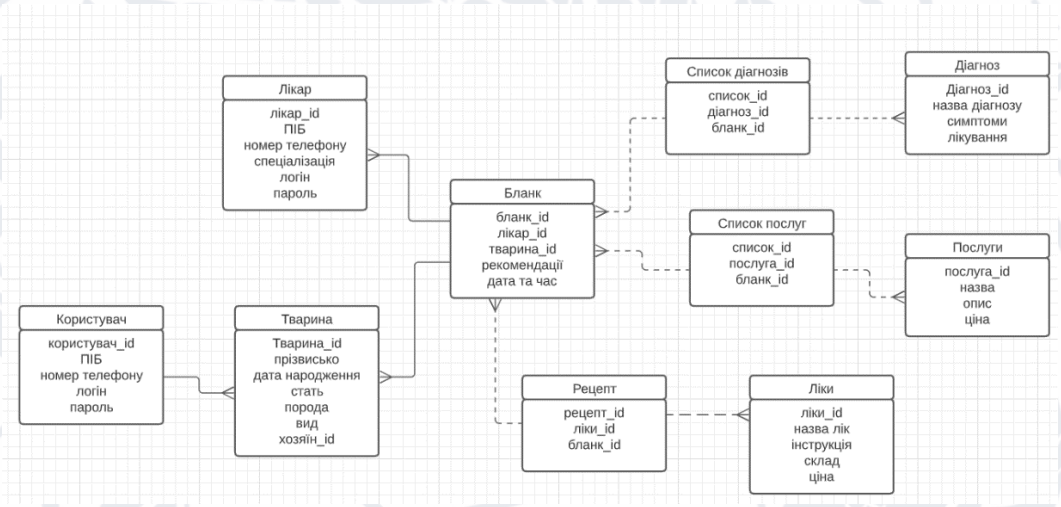


Рисунок 2.1 – ER-діаграма за нотацією Баркера

Для створення логічної моделі бази даних важливо досягти третьої нормальної форми, забезпечивши наступні вимоги:

1. Кожна таблиця повинна містити первинний ключ.
2. Кожен атрибут повинен мати лише одне значення.
3. Кожен атрибут, що не входить у ключ, повністю залежить від первинного ключа відношення.
4. Між атрибутами, які не входять у ключ, не повинно бути транзитивних залежностей.

Розпочнемо з розгляду сутності "Лікар". Відповідно до цього, створимо відношення "doctor", яке має наступні атрибути: ПІБ (doctor_name), номер телефону (phone_num), спеціалізація (Specialization), логін та пароль (login, pass), а також первинний ключ (doctor_id).

Далі, розглянемо сутність "Користувач" і побудуємо відношення "user" з

атрибутами: первинний ключ (id), ПІБ (name), номер телефону (phone), логін та пароль (login, pass).

Перейдемо до сутності "Діагноз" і сформуємо відношення "diagnosis" із наступними атрибутами: первинний ключ (diag_id), назва (name), симптоми (symptom), лікування (treatment).

Сутність "Ліки" визначимо як відношення "medications" з атрибутами: первинний ключ (Med_Id), назва (Name), інструкція застосування (Instruction), склад (Composition), кількість (Amount), ціна (price).

Для сутності "Послуги" створимо відношення "services" із наступними атрибутами: первинний ключ (service_id), назва (name), опис (description), ціна (price). Нарешті, розглянемо сутність "Прийом" та визначимо відношення "form" з атрибутами: "doctor_id", "pass_id", "diagnosis_id", "drug_id", рекомендації (recommendations), дата та час (data_time), та первинний ключ "form_number".

Схему бази даних продемонстровано на рис. 2.2. Вивчивши цю структуру, можна стверджувати, що вона відповідає всім критеріям третьої нормальної форми.

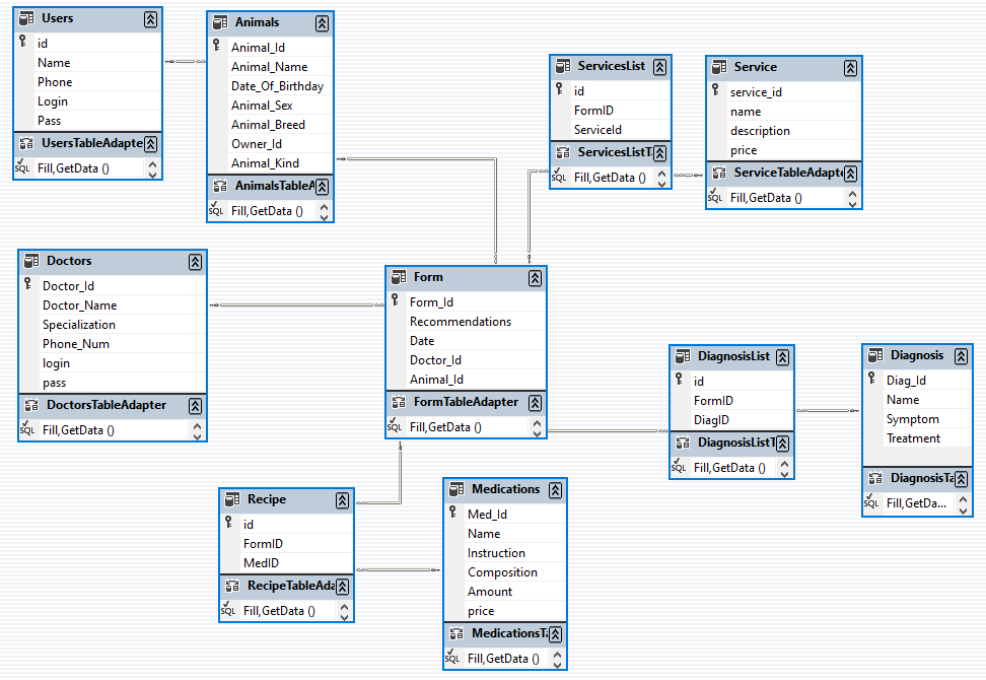


Рисунок 2.2 – Схема бази даних

## 2.2 Середовище розробки Visual Studio та мова програмування C#

C# – це мова програмування загального призначення високого рівня, яка підтримує кілька парадигм. C# охоплює статичну типізацію [16] сильну типізацію, лексичну область видимості, імперативну, декларативну, функціональну, загальну [16] об’єктно-орієнтоване (на основі класів) і компонентно-орієнтоване програмування дисципліни.[17]

Мова програмування C# була створена Андерсом Хейлсбергом у компанії Microsoft у 2000 році, отримала статус міжнародного стандарту Ecma у 2002 році та ISO/IEC у 2003 році. Первісно C# була представлена разом із .NET Framework і Visual Studio, які були закритими продуктами, оскільки Microsoft тоді не використовувала відкритий код. Проте у 2004 році стартував відкритий проект Mono, який надавав кросплатформенний компілятор і середовище виконання для C#. Спустя десять років Microsoft випустила безкоштовні продукти з відкритим кодом, такі як Visual Studio Code (редактор коду), Roslyn (компілятор) і уніфікована платформа .NET (фреймворк програмного забезпечення), які підтримують C# і працюють на різних платформах. Хоча Mono приєднався до Microsoft, він залишився окремим від .NET.

Стандарт Ecma перераховує такі цілі дизайну для C# [17]:

1. Ця мова має бути простою, сучасною, об’єктно-орієнтованою мовою програмування загального призначення.

2. Мова і її реалізації повинні підтримувати основні принципи розробки програмного забезпечення, такі як сильна перевірка типів, контроль меж масивів, виявлення спроб використання неініціалізованих змінних і автоматичне прибирання сміття. Забезпечення міцності, довговічності і продуктивності для програміста має важливе значення.

3. Мова програмування розроблена з метою створення програмних компонентів, які можуть бути успішно впроваджені в розподілених середовищах.

Її основна спрямованість полягає в створенні програм, які демонструють високу придатність для ефективного функціонування в умовах розподіленості. Такий підхід дозволяє розробникам створювати компоненти, які легко і ефективно взаємодіють у розподілених мережевих середовищах, що є важливим в контексті сучасних технологій розробки програмного забезпечення.

4. Переносимість має велике значення для вихідного коду і зручності для програмістів, особливо для тих, хто має досвід роботи з C та C++.

5. Підтримка інтернаціоналізації[20] дуже важлива.

6. Мова програмування C# призначена для розробки програм як для розподілених, так і вбудованих систем. Це може бути від дуже великих систем, що використовують складні операційні системи, до дуже малих систем із спеціальними функціями.

7. Незважаючи на те, що додатки C# мають бути економічними щодо вимог до пам’яті та процесорної потужності, ця мова не була призначена для прямої конкуренції за продуктивністю та розміром із C або мовою асемблера.[21]

Під час створення бібліотеки класів .NET Framework спочатку використовувалася система компілятора керованого коду під назвою «Simple Managed C» (SMC) [22, 23]. У січні 1999 року Андерс Хейлсберг утворив команду для розробки нової об’єктно-орієнтованої мови під назвою Cool, що означає «C-подібна об’єктно-орієнтована мова» [24]. Компанія Microsoft розглядала можливість залишити "Cool" як остаточну назву, але вирішила змінити її з міркувань торгової марки. Коли проект .NET був публічно представлений на конференції професійних розробників у липні 2000 року, мову перейменували в C#, а бібліотеки класів і середовище виконання ASP.NET були перенесені на C#.

Джеймс Гослінг, творець мови програмування Java (створеної у 1994 році), та Білл Джой, співзасновник Sun Microsystems і один з авторів Java, визнали C# «імітацією» Java. Пізніше Гослінг висловив думку, що «C# – це варіація Java, але

без надійності, продуктивності та безпеки» [26, 27]. У липні 2000 року Хейлсберг відзначив, що C# «не є копією Java» і має «набагато більше спільного з C++» у своєму дизайні [28].

Починаючи з моменту випуску C# 2.0 у листопаді 2005 року, мови C# та Java почали рухатися в різних напрямках, перетворившись у дві повністю відмінні мови. Однією зі значущих різниць став додаток генериків до обох мов із відмінними реалізаціями. C# використовує реифікацію для забезпечення "першокласних" загальних об'єктів, які можна використовувати так, як будь-який інший клас, із генерацією коду, що виконується при завантаженні класу. До того ж C# додав низку важливих можливостей для програмування у функціональному стилі. Відзначимо LINQ, що вийшов разом із C# 3.0, та супутній набір лямбда-виразів, методів розширення та анонімних типів. Ці функції дають програмістам C# можливість використовувати методи функціонального програмування, такі як замикання, коли це виправдано. LINQ і функціональний імпорт допомагають розробникам зменшити обсяг шаблонного коду, необхідного для таких загальних завдань, як опитування баз даних або аналіз файлів XML, зосереджуючись на самій логіці програми для покращення читабельності та обслуговуваності.

У C# був талісман Енді (названий на честь Андерса Хейлсберга). Він був знятий з експлуатації 29 січня 2004 року.[32]

Корпорація Майкрософт вперше застосувала термін "C#" у 1988 році для версії мови C, призначеної для інкрементної компіляції. Однак цей конкретний проект не був завершений, і назва "C#" була використана знову лише на пізніших етапах.

Назва «до-дієз» була натхненна нотним записом, згідно з яким символ дієзу вказує на те, що написана нота має бути на півтону вищою за висотою.[38] Це схоже на назву мови C++, де "++" вказує на те, що змінна має бути збільшена на 1 після оцінки. Знак різкості (C#) також візуально вказує на лігатуру, що

складається з чотирьох символів «+», утворюючи сітку два на два. Це може додатково сигналізувати, що мова представляє собою розвиток мови C++. [39]

Через технічні обмеження відображення (стандартні шрифти, браузери тощо) і той факт, що символ різкості (U+266F ♯ МУЗИЧНИЙ ЗНАК SHARP (♯)) відсутній на більшості розкладок клавіатури, знак цифри (U+0023 # ЗНАК ЧИСЛА (#)) було обрано для наближення символу різкості в письмовій назві мови програмування.[40] Ця конвенція відображена в Специфікації мови C# ECMA-334.[17]

Суфікс «sharp» використовувався низкою інших мов .NET, які є варіантами існуючих мов, включаючи J# (мову .NET, також розроблену Microsoft, яка є похідною від Java 1.1), A# (від Ada) і мова функціонального програмування F#.[41] Початкова реалізація Eiffel для .NET називалася Eiffel#[42], назва вилучена, оскільки тепер підтримується повна мова Eiffel. Суфікс також використовувався для таких бібліотек, як Gtk# (обгортка .NET для GTK та інших бібліотек GNOME) і Cocoa# (обгортка для Cocoa).

C# був задуманий як мова програмування, яка найбільш прямо відображає базову інфраструктуру спільної мови (CLI).[68] Багато внутрішніх типів C# відповідають типам значень, які реалізовані у фреймворку CLI. Однак специфікація мови не визначає конкретні вимоги до генерації коду. З іншого боку, вона не зобов'язує компілятор C# цілісно націлювати на середовище виконання спільної мови, генерувати код проміжної мови (CIL), або використовувати будь-який конкретний формат. Таким чином, деякі компілятори C# можуть навіть генерувати машинний код, подібний до того, який генерують традиційні компілятори C++ чи Fortran. [69, 70]

C# підтримує суворе, неявно типізоване оголошення змінних за допомогою ключового слова var [16] та неявно типізовані масиви з ключовим словом new[] з наступним ініціалізатором колекції.[16, 20].

C# володіє строгим логічним типом даних - bool. Конструкції, які використовують умови, такі як `while` і `if`, потребують виразу типу, який реалізує логічні операції, наприклад, типу `Boolean`. Навіть якщо в C++ також є логічний тип, його можна вільно конвертувати в цілі числа і навпаки, і вирази, такі як `if (a)`, вимагають лише, щоб `a` було конвертовано в `bool`, дозволяючи `a` бути, наприклад, цілим числом або вказівником. У C# цей підхід "ціле число, яке представляє істину чи хибність" заборонений. Такий підхід може викликати певні типи помилок у програмі, наприклад, якщо використовується `if (a = b)` замість `if (a == b)`, тому що перше - це присвоєння, а не порівняння.

C# вважається більш безпечною мовою порівняно з C++. Її підхід до неявних перетворень є консервативним: тільки ті, які вважаються безпечними, наприклад, розширення цілих чисел, є дозволеними за замовчуванням. Ці перетворення обов'язково виконуються під час компіляції, JIT (Just-In-Time Compilation) і, у деяких випадках, під час виконання.

C# не дозволяє неявних перетворень між логічними значеннями та цілими числами, а також між членами перерахування та цілими числами (за винятком літералу 0, який може бути неявно перетворений на будь-який перерахований тип). Будь-яке перетворення, визначене користувачем, повинно бути явно позначеним як явне або неявне, на відміну від C++, де конструктори копіювання та оператори перетворення за замовчуванням є неявними.

У C#, на відміну від C++, здійснена явна підтримка коваріантності та контраваріантності у загальних типах. Це означає, що можна безпечно використовувати більш конкретний тип, ніж очікується, або менш конкретний тип, де відповідно. Наприклад, в C# можливо використовувати базовий тип, де очікується похідний, або використовувати більш конкретний тип там, де очікується більш загальний.

У C++, підтримка контраваріантності реалізується в обмеженому вигляді

через семантику типів повернення віртуальних методів.

C# відрізняється тим, що вона не підтримує глобальних змінних або функцій. Усі функції та члени повинні бути оголошені в межах класів. Статичні члени публічних класів можуть функціонувати як заміна глобальних змінних та функцій.

Важливо зауважити, що локальні змінні в C# не можуть закривати змінні охоплюючого блоку, це відрізняється від мов програмування C і C++.

Платформа компілятора .NET, відома як Roslyn, відкриває доступ до API для управління процесом компіляції мови, дозволяючи компілювати код на C# з програм .NET. Це API включає можливості синтаксичного (лексичного) аналізу коду, семантичного аналізу, динамічної компіляції в CIL і видачі коду[73].

За допомогою функцій компілятора Roslyn для C#, розробники можуть використовувати метапрограмування під час часу компіляції. Під час процесу компіляції вони мають можливість перевірити код, що компілюється, використовуючи API компілятора та передати додатковий згенерований вихідний код C# для подальшої компіляції.

В C#, метод - це член класу, що може бути викликаний як функція (послідовність інструкцій) і не просто утримує значення властивості класу. Як і в інших мовах схожого синтаксису, таких як C++ та ANSI C, сигнатура методу складається з наступних елементів: будь-які ключові слова доступності (наприклад, private), явна специфікація його типу повернення (наприклад, int або ключове слово void, якщо значення не повертається), ім’я методу, та послідовність параметрів, розділених комами і укладених у дужках. Кожен параметр включає тип, офіційне ім'я та, за бажанням, значення за замовчуванням, що використовується, якщо жодне не надано. Деякі конкретні типи методів, такі як ті, що просто отримують або встановлюють властивість класу через повернення значення або призначення, не вимагають повного оголошення

підпису, але в загальному випадку визначення класу включає повне оголошення підпису його методів.

Аналогічно C++ і в відміну від Java, у програмістів C# повинно бути ключове слово-модифікатора області видимості `virtual`, щоб дозволити підкласам заміщувати методи.

Методи розширення в C# дають можливість програмістам використовувати статичні методи, подібно до того, як вони є частиною таблиці методів класу. Ця можливість дозволяє додавати методи до об’єктів так, начебто ці методи вже існують у цих об’єктах або їхніх похідних.

Механізм динамічного типу в C# дозволяє зв'язувати методи під час виконання. Це відкриває можливості для викликів методів, схожих на ті, які використовуються в JavaScript, а також для динамічної композиції об'єктів під час виконання. [16]

В C# існує підтримка строго типізованих покажчиків на функції через ключове слово "делегат". Ця особливість подібна до концепції сигналів та слотів у фреймворку Qt для псевдо-C++. Однак в C# використовується семантика, що спеціально оточує події в стилі публікації-підписки, а для цього використовуються саме делегати.

C# пропонує синхронізовані виклики методів, подібні до Java, за допомогою атрибута [MethodImpl(MethodImplOptions.Synchronized)] і підтримує взаємовиключні блокування за допомогою ключового слова lock.

C# надає підтримку класів з властивостями, які можуть мати прості функції доступу з резервним полем або реалізовувати функції отримання та встановлення.

Від версії C# 3.0 введено синтаксичний цукор для автоматично реалізованих властивостей, де аксесор (getter) і мутатор (setter) інкапсулюють операції над одним атрибутом класу.

Простір імен у мові C# забезпечує аналогічний рівень ізоляції коду, що і пакет у Java або простір імен у C++, і володіє дуже схожими правилами та функціональністю. Простори імен можна імпортувати за допомогою синтаксису "використовуючи".[77]

У мові програмування C#, можна використовувати покажчики адрес пам'яті лише в спеціально позначених блоках коду як небезпечні [78]. Для програм, що включають небезпечний код, необхідні відповідні дозволи для виконання. Зазвичай доступ до об'єктів здійснюється через безпечні посилання на об'єкти, які завжди вказують на "живий" об'єкт або мають чітко визначене нульове значення. Отримати посилання на "мертвий" об'єкт (тобто той, що був зібраний системою сміття) чи на випадковий блок пам'яті не можливо.

Небезпечні покажчики можуть вказувати на екземпляри некерованих типів значень, таких як екземпляри класів, масиви або рядки, які не містять посилань на об'єкти, піддаються системі сміття. Код, який не визначено як небезпечний, все ще може зберігати і маніпулювати покажчиками через тип System.IntPtr, але не може їх розіменовувати.

Керовану пам'ять у мові C# неможливо явно вивільнити; замість цього використовується автоматичне збирання сміття. Цей процес вирішує проблему витоків пам'яті, оскільки відповідальність за вивільнення пам'яті відводиться від програміста. Він дозволяє автоматично вивільняти пам'ять, яка в більшості випадків більше не потрібна. Хоча код, який зберігає посилання на об'єкти довше, ніж це дійсно необхідно, може споживати більше пам'яті, ніж необхідно, проте, коли останнє посилання на об'єкт звільнюється, пам'ять стає доступною для автоматичного збирання сміття.

Програмістам відкритий асортимент стандартних виняткових ситуацій. Функції, що входять до стандартних бібліотек, регулярно можуть викидати винятки за деяких обставин, і діапазон викиданих винятків зазвичай

документується. Спеціальні класи винятків можуть бути створені для ситуацій, де потрібно виконати обробку для конкретних умов.[79]

В C# відсутні перевірені винятки (на відміну від Java), що є свідомим рішенням, зумовленим проблемами масштабованості та версійності. Це відрізняється від C++, оскільки C# не підтримує множинне успадкування, але дозволяє класам реалізовувати будь-яку кількість "інтерфейсів" (повністю абстрактних класів). Таке рішення прийняте для уникнення ускладнень та спрощення архітектурних вимог в рамках CLI.

У C#, коли реалізується кілька інтерфейсів, які включають методи з однаковою назвою та параметрами одного типу в однаковому порядку (тобто мають однакову сигнатуру), є можливість вибору між тим, чи охопити всі ці методи одним методом, чи, у випадку потреби, створити окремі методи для кожного інтерфейсу.

У порівнянні з Java, в C# існує підтримка перевантаження операторів. Це дозволяє визначати власні варіанти роботи операторів для користувач в межах мови.

Також, в C# існує можливість використовувати LINQ за допомогою .NET Framework. Розробник може опитувати різноманітні джерела даних, якщо об'єкт реалізований інтерфейсом IEnumerable<T>. Це охоплює XML-документи, набори даних ADO.NET і бази даних SQL.

Використання LINQ у C# надає численні переваги, такі як підтримка Intellisense, потужні можливості фільтрації, безпека типів і можливість перевірки помилок компіляції, а також узгодженість для ведення запитів до даних із різних джерел.

Додатково, в C# існує уніфікована система типів, відома як Common Type System (CTS). Ця система передбачає, що всі типи, навіть примітивні, такі як цілі числа, є підкласами класу System.Object. Наприклад, всі типи успадковують

метод ToString().

Екземпляри типів значень не мають ані посилальної ідентичності, ані посилальної семантики порівняння. Порівняння рівності та нерівності для типів значень порівнює фактичні значення даних у примірниках, якщо відповідні оператори не перевантажені. Типи значень походять від System.ValueType, завжди мають значення за замовчуванням і завжди можуть бути створені та скопійовані. Інші обмеження на типи значень включають в себе те, що вони не можуть успадковувати один від одного (але можуть реалізовувати інтерфейси) і не можуть мати явного конструктора за замовчуванням (без параметрів), оскільки вони вже мають неявний, який ініціалізує всі дані, що містяться, залежним від типу значенням за замовчуванням (0, нуль або подібне). Прикладами типів значень є всі примітивні типи, такі як `int` (32-розрядне ціле число зі знаком), `float` (32-розрядне число з плаваючою комою IEEE), char (16-розрядний код Unicode) і System.DateTime ( визначає конкретний момент часу з точністю до наносекунд). Іншими прикладами є enum (перерахування) і struct (визначені користувачем структури).

Навпаки, посилальні типи мають поняття посилальної ідентичності, тобто кожен екземпляр посилального типу за своєю суттю відрізняється від будь-якого іншого екземпляра, навіть якщо дані в обох екземплярах однакові. Це виявляється у порівняннях на рівність та нерівність для еталонних типів, які визначають рівність на основі посилань, а не структурну рівність, якщо тільки відповідні оператори не перевизначені (наприклад, у випадку System.String). Деякі операції не завжди можливі, наприклад, створення екземпляру еталонного типу, копіювання існуючого екземпляра або виконання порівняння значень двох існуючих екземплярів. Хоча певні типи посилань можуть надавати такі послуги, відкриваючи відкритий конструктор або реалізовуючи відповідний інтерфейс .

Обидві категорії типів можна розширити за допомогою типів, визначених

користувачем.

Боксинг в C# — це процес неявного перетворення об'єкта типу значення у відповідне еталонне значення.

Розпакування – це операція перетворення значення посилального типу (раніше упакованого в коробку) у значення типу значення.[94] Розпакування в C# вимагає явного приведення типу. Коробковий об’єкт типу T може бути лише розпакований до T (або T з можливістю нульового значення).[95]

Мову C# і стандарт Common Language Infrastructure (CLI) було стандартизовано відповідно до стандартів ISO/IEC і Ecma. Це сприяє розумному і недискримінаційному захисту ліцензування від патентних претензій.

Майкрософт спочатку взялася утриматися від судових позовів стосовно порушення патентів у некомерційних проектах, якщо вони використовують частину фреймворку, яка охоплена Open Specification Promise.[99] Корпорація Майкрософт також погодилася не застосовувати патенти, пов’язані з продуктами Novell, проти клієнтів Novell, які платять [100]. Проте Novell стверджувала, що Mono не порушує жодних патентів Microsoft.[102] Корпорація Майкрософт також уклала спеціальну угоду не застосовувати патентні права, пов’язані з плагіном браузер Moonlight, який ґрунтується на Mono, якщо цей проект був отриманий через Novell. Це було частиною стратегії з прозорішості та співпраці зі спільнотою відкритого програмного забезпечення. [103]

Протягом останніх десяти років Microsoft розширила свій підхід до відкритого програмного забезпечення та кросплатформеності. Компанія випустила безкоштовні та відкриті інструменти для розробки на C#, такі як Visual Studio Code, .NET Core і Roslyn. Цей крок дозволяє розробникам використовувати C# на різних платформах та в різних середовищах розробки, забезпечуючи більшу гнучкість та доступність.

Крім того, придбання Xamarin дозволило Microsoft активно підтримувати

розробку мобільних додатків на C# для різних платформ, включаючи iOS та Android. Такий підхід дозволяє розробникам використовувати C# для розробки не лише настільних, а й мобільних застосунків, розширюючи сферу застосування мови програмування.

Ці кроки свідчать про стратегічний поворот Microsoft у напрямку відкритого програмного забезпечення та кросплатформеності, надаючи розробникам більше можливостей та ресурсів для створення програмного забезпечення.

Корпорація Майкрософт активно розвиває довідкові компілятори для мови програмування C# із відкритим вихідним кодом, представивши два ключові інструменти. Перший з них, компілятор з назвою Roslyn, призначений для компіляції в проміжну мову (IL), а другий, RyuJIT, є JIT-компілятором (Just-In-Time), що працює динамічно. RyuJIT відзначається здатністю виконувати оптимізації на льоту та конвертує IL-код в оптимізований машинний код для інтерфейсу ЦП.[105] RyuJIT є відкритим кодом і написаний на C++.[106] Roslyn повністю написаний у керованому коді (C#), був відкритий і функціональні можливості з’явилися як API. Таким чином, це дозволяє розробникам створювати інструменти рефакторингу та діагностики.[4][107] Існує дві гілки офіційної реалізації мови: .NET Framework, яка має закритий код і призначена лише для операційних систем Windows, і .NET Core, що має відкритий вихідний код і є крос-платформенною. Згодом ці гілки були об'єднані в одну реалізацію з відкритим кодом під назвою .NET 5.0. В .NET Framework 4.6 також був введений новий JIT-компілятор, який замінив попередню версію.

Інші компілятори C# (деякі з яких містять реалізацію спільної мовної інфраструктури та бібліотек класів .NET):

Mono, спонсорований Microsoft проект, надає компілятор C# з відкритим вихідним кодом, повну реалізацію CLI з відкритим кодом (включно з

необхідними бібліотеками рамки, як вони вказані в специфікації ECMA) і майже повну реалізацію бібліотек класів NET. до .NET Framework 3.5.

Платформа інструментів Elements від RemObjects включає в себе RemObjects C#, який перетворює код, написаний мовою C#, в байт-код .NET Common Intermediate Language, Java, Cocoa, байт-код Android, WebAssembly, а також в рідний машинний код для операційних систем Windows, macOS і Linux.

Проект DotGNU, хоча тепер вже припинений, колись надавав компілятор C# з відкритим вихідним кодом. Також в рамках цього проекту розроблялася майже повна реалізація інфраструктури спільної мови, включаючи необхідні бібліотеки рамки, як вони вказані в специфікації ECMA. Крім того, DotGNU містив підмножину власних бібліотек Microsoft .NET, включаючи ті, які не були задокументовані та не входили до специфікації ECMA, але входили до стандартного дистрибутива Microsoft .NET Framework до версії 2.0.

Unity використовує C# як основну мову сценаріїв у своєму ігровому движку. Щодо ігрового движка Godot, важливо зауважити, що до нього був доданий додатковий модуль для підтримки мови C#. Цей модуль був реалізований завдяки пожертві в розмірі 24 000 доларів від компанії Microsoft.

Visual Studio — це інтегроване середовище розробки, яке містить ряд корисних інструментів для програмістів. Редактор коду включає IntelliSense, що допомагає автоматично доповнювати код, і функції рефакторингу для полегшення оптимізації коду. Вбудований налагоджувач працює на рівні джерела і машини, спрощуючи виявлення та виправлення помилок.

Серед інших інструментів можна виділити профайлер коду для аналізу продуктивності, конструктор для розробки графічних інтерфейсів користувача, веб-дизайнер, конструктор класів та інструменти для роботи з базами даних. Visual Studio також підтримує плагіни, які розширюють його можливості, такі як підтримка різних систем контролю версій (наприклад, Subversion і Git) і

додавання нових інструментів для розробки програмного забезпечення.

Найпростіша версія Visual Studio, версія Community, доступна безкоштовно. Слоган видання Visual Studio Community: «Безкоштовна, повнофункціональна IDE для студентів, розробників із відкритим кодом і індивідуальних розробників».

Станом на 10 січня 2023 року Visual Studio 2022 є поточною виробничою версією. Visual Studio 2013, 2015 і 2017 мають розширену підтримку, тоді як 2019 – основну підтримку.[9]

Visual Studio не прив'язаний до жодної конкретної мови програмування, рішення або інструменту. Натомість він дозволяє підключати функціональність, що закодована як VSPackage. Після встановлення функціональність стає доступною як послуга. Інтегроване середовище розробки (IDE) надає три основні служби: SVsSolution, яка дозволяє перераховувати проекти та рішення; SVsUIShell, що забезпечує роботу вікон і функціональність інтерфейсу користувача, включаючи вкладки, панелі інструментів і вікна інструментів; і SVsShell, яка відповідає за реєстрацію VSPackages. Усі редактори, дизайнери, типи проектів та інші інструменти реалізовані як VSPackages. Використовуючи COM, Visual Studio взаємодіє з VSPackages. SDK Visual Studio також включає керовану структуру пакетів (MPF), яка представляє собою набір керованих оболонок для COM-інтерфейсів і дозволяє писати пакети будь-якою мовою, сумісною з CLI. Однак MPF не включає всі можливості, доступні через COM-інтерфейси Visual Studio. Ці служби можна використовувати для створення додаткових пакетів, які розширюють функціональність середовища розробки Visual Studio.

Для додавання підтримки різних мов програмування використовується спеціальний VSPackage, відомий як "Language Service". Мовний сервіс визначає різні інтерфейси, які можуть бути реалізовані в рамках VSPackage для додавання

підтримки різних функцій. Ці функціональності включають забарвлення синтаксису, автодоповнення операторів, вирівнювання фігурних дужок, виведення підказок із інформацією про параметри, а також списки членів і маркери помилок для фонової компіляції.[13] Якщо інтерфейс реалізовано, функціональність буде доступна для мови. Мовні служби реалізуються на основі кожної мови. Різні реалізації можуть використовувати код із синтаксичного аналізатора або компілятора для конкретної мови. [13] Мовні служби можуть бути реалізовані як у рідному коді, так і в керованому коді. Для рідного коду можна використовувати або рідні COM-інтерфейси, або Babel Framework (частина Visual Studio SDK).[14] Для керованого коду MPF містить оболонки для написання керованих мовних служб.[15]

Visual Studio не включає в себе вбудованої підтримки систем керування версіями, проте визначає два альтернативні методи інтеграції систем керування версіями з Інтегрованою середовищею розробки (IDE).[16] Source Control VSPackage може надавати власний налаштований інтерфейс користувача. Навпаки, плагін керування джерелом, що використовує MSSCCI (інтерфейс керування вихідним кодом Microsoft), надає набір функцій, які використовуються для реалізації різноманітних функціональних можливостей керування джерелом за допомогою стандартного інтерфейсу користувача Visual Studio. [17,18]

Visual Studio підтримує одночасний запуск кількох екземплярів середовища, кожен з власним набором VSPackages. Кожен екземпляр використовує відокремлені кущі реєстру, (див. визначення MSDN терміну «кущ реєстру» у значенні, яке тут використовується) для збереження свого стану конфігурації та розрізняються за їхнім AppId (ідентифікатором програми). Екземпляри запускаються спеціальним .exe для AppId, який вибирає AppId, встановлює кореневий кущ. Різні версії продукту Visual Studio створюються з використанням різних AppId. Продукти випуску Visual Studio Express

встановлюються з власними ідентифікаторами додатків, але продукти Standard, Professional і Team Suite мають однаковий ідентифікатор додатків. Отже, випуски Express можна інсталювати поруч з іншими випусками, на відміну від інших випусків, які оновлюють ту саму установку. Професійна версія включає надмножину пакетів VSPackages у стандартній версії, а груповий пакет включає надмножину пакетів VSPackages в обох інших версіях. Система AppId використовується оболонкою Visual Studio Shell у Visual Studio 2008.[19]

Visual Studio містить редактор коду, який підтримує підсвічування синтаксису та завершення коду за допомогою IntelliSense для змінних, функцій, методів, циклів і запитів LINQ.[20] Під час розробки веб-сайтів і веб-додатків, IntelliSense включає підтримку для включених мов, а також для XML, каскадних таблиць стилів і JavaScript.[21, 22] Пропозиції автозавершення з’являються в немодальному полі списку над вікном редактора коду, поруч із курсором редагування. У Visual Studio 2008 і новіших версіях його можна тимчасово зробити напівпрозорим, щоб побачити код, який він закриває.[20] Редактор коду використовується для всіх підтримуваних мов.

Редактор коду у Visual Studio також підтримує встановлення закладок у коді для швидкої навігації. Інші навігаційні засоби включають згортання блоків коду та інкрементний пошук, на додаток до звичайного текстового пошуку та пошуку за регулярними виразами.[23] Редактор коду включає багатоелементний буфер обміну та список завдань. Фрагменти коду, що є збереженими шаблонами для повторюваного коду, підтримуються як редактором коду. Ці фрагменти можна вставляти в код і налаштовувати для поточного проекту. Відділено від цього, вбудований інструмент керування дозволяє ефективно використовувати фрагменти коду. Інструменти відображаються у вигляді плаваючих вікон, які можна автоматично приховувати, коли не використовуються, або прикріпляти збоку до екрана. Редактор коду в Visual Studio також обладнаний функціоналом

рефакторингу коду, таким як зміна порядку параметрів, перейменування змінних і методів, вилучення інтерфейсів та інкапсуляція членів класу всередині властивостей, серед іншого.

Налагоджувач в Visual Studio функціонує як на рівні вихідного коду, так і на рівні машинного коду. Він взаємодіє як з керованим кодом, так і з рідним кодом, і може бути використаний для налагодження програм, написаних будь-якою мовою, яку підтримує Visual Studio. Крім того, він також може приєднуватися до запущених процесів, контролювати та налагоджувати ці процеси.[24] Якщо вихідний код для запущеного процесу доступний, він відображає код під час виконання. Якщо вихідний код недоступний, налагоджувач Visual Studio може відобразити розбирання програми. Крім того, він може створювати дампи пам'яті та завантажувати їх для подальшого налагодження. Підтримується також налагодження багатопотокових програм, і можливо налаштувати налагоджувач на автоматичний запуск у випадку аварійного завершення роботи програми, що виконується поза середовищем Visual Studio.

Налагоджувач Visual Studio дозволяє встановлювати точки зупину (які дозволяють тимчасово зупинити виконання в певній позиції) і спостереження (які відстежують значення змінних під час виконання).[26] Код можна переступати, тобто виконувати один рядок (вихідного коду) за раз. Він може або переходити до функцій для налагодження всередині неї, або переходити через неї, тобто виконання тіла функції недоступне для перевірки вручну.[27] Під час налагодження, якщо покажчик миші навести на будь-яку змінну, її поточне значення відображається у спливаючій підказці («спливаючих підказках даних»). Під час кодування налагоджувач Visual Studio дозволяє викликати певні функції вручну з вікна інструменту Imediate. Параметри методу надаються у вікні Immediate.[28]

Visual Studio містить безліч візуальних дизайнерів, які допомагають у розробці програм. Ці інструменти включають:

Дизайнер Windows Forms використовується для створення програм графічного інтерфейсу за допомогою Windows Forms. Макетом можна керувати, помістивши елементи керування в інші контейнери або заблокувавши їх збоку від форми. Елементи керування, які відображають дані (наприклад, текстове поле, поле зі списком і сітку), можна прив’язати до джерел даних, таких як бази даних або запити. Елементи керування з прив’язкою до даних можна створити, перетягнувши елементи з вікна «Джерела даних» на поверхню дизайну.[29] Інтерфейс користувача пов’язаний із кодом за допомогою моделі програмування, керованої подіями. Дизайнер генерує код C# або VB.NET для програми.

Дизайнер WPF, під кодовою назвою Cider [30], був введений у Visual Studio 2008. Аналогічно до дизайнера Windows Forms, він підтримує метафору перетягування для створення інтерфейсів користувача, спрямованих на Windows Presentation Foundation. Цей інструмент обладнаний всіма можливостями WPF, включаючи зв'язування даних та автоматичне управління макетом. В результаті генерує код XAML для інтерфейсу користувача. Створений файл XAML сумісний із Microsoft Expression Design, який є продуктом, орієнтованим на дизайнерів. Код XAML пов'язується з кодом за допомогою моделі коду.

Visual Studio включає редактор і дизайнер веб-сайтів, що надає можливість створювати веб-сторінки шляхом перетягування віджетів. Цей інструмент застосовується для розробки програм ASP.NET і підтримує мови HTML, CSS і JavaScript. Він використовує модель коду для зв’язку з кодом ASP.NET. З Visual Studio 2008 і далі, механізм макета, використовуваний веб-дизайнером, є спільним із припиненою версією Expression Web. Додатково, надається підтримка ASP.NET MVC як окремого завантаження[31], а також проект ASP.NET Dynamic Data, наданого корпорацією Microsoft.[32]

Конструктор класів використовується для створення та редагування класів (включно з його членами та їхнім доступом) .

Засіб для графічного редагування схем бази даних, включаючи таблиці з типізацією, первинні та зовнішні ключі та обмеження, називається конструктором даних. Він також придатний для розробки запитів у графічному вигляді.

Починаючи з Visual Studio 2008, конструктор зіставлення використовується LINQ to SQL для розробки зіставлення між схемами бази даних і класами, які інкапсулюють дані. Нове рішення від підходу ORM, ADO.NET Entity Framework, замінює та покращує стару технологію.

Інструмент «Редактор властивостей» використовується для редагування властивостей на панелі GUI у Visual Studio. Він містить список усіх доступних властивостей (як доступних лише для читання, так і тих, які можна встановити) для всіх об’єктів, включаючи класи, форми, веб-сторінки та інші елементи.

Data Explorer використовується для керування базами даних на екземплярах Microsoft SQL Server. Він дозволяє створювати та змінювати таблиці бази даних (за допомогою команд T-SQL або за допомогою дизайнера даних). Його також можна використовувати для створення запитів і збережених процедур, причому останні в T-SQL або в керованому коді через SQL CLR. Також доступна підтримка налагодження та IntelliSense.

Інструмент Server Explorer використовується для керування підключеннями до бази даних на доступному комп’ютері. Він також використовується для перегляду запущених служб Windows, лічильників продуктивності, журналу подій Windows і черг повідомлень і використання їх як джерела даних.[35]

Windows Forms (WinForms) – це безкоштовна бібліотека графічних (GUI) класів для настільних ПК, ноутбуків і планшетів. ПК.[2] Хоча вона розглядається

як заміна попередньої та більш складної.

На події Microsoft Connect, яка відбулася 4 грудня 2018 року, Microsoft оголосила про випуск проекту Windows Forms як відкритий код на GitHub. Цей випуск відбувся під ліцензією MIT і дозволив проектам, спрямованим на платформу .NET Core, використовувати Windows Forms. Важливо відзначити, що хоча Windows Forms стала доступною для проектів .NET Core, фреймворк залишається призначеним виключно для платформи Windows. Крім того, кросплатформна реалізація у Mono залишається єдиною альтернативою в цьому відношенні.

Програма Windows Forms - це західна програма, орієнтована на події, і підтримується від Microsoft .NET Framework. Відмінною рисою є те, що вона витрачає значну частину свого часу, очікуючи подій від користувача, таких як заповнення текстового поля або натискання кнопки. Код програми можна розробляти на мові програмування .NET, такій як C# або Visual Basic.

Windows Forms надає можливість використовувати стандартні елементи керування інтерфейсу користувача Windows, перетворюючи існуючий API Windows у керований код. Завдяки Windows Forms у .NET Framework стає доступною більш повна абстракція над Win32 API, порівняно з Visual Basic або MFC.

Windows Forms подібний до бібліотеки Microsoft Foundation Class (MFC) у відносинах з розробкою клієнтських програм, але надає оболонку на основі класів C++, спрощуючи створення програм для Windows. У порівнянні з MFC, Windows Forms не встановлює стандартну структуру додатків. В кожному елементі керування програми Windows Forms міститься екземпляр конкретного класу.

Усі візуальні елементи в бібліотеці класів Windows Forms успадковують від класу Control. Це надає базовий набір функціональностей для елементів

інтерфейсу користувача, таких як розміщення, розмір, колір, шрифт, текст, а також основні події, наприклад, клацання та перетягування. Клас Control також підтримує стикування, що дозволяє елементам керування змінювати своє положення відносно батьківського елемента. Технологія Microsoft Active Accessibility у класі Control полегшує використання Windows Forms користувачами із обмеженими можливостями.

У Visual Studio форми створюються за допомогою техніки перетягування. Інструмент використовується для розміщення елементів керування (наприклад, текстових полів, кнопок тощо) на формі (вікні). Елементи керування мають пов’язані з ними атрибути та обробники подій. Значення за замовчуванням надаються під час створення елемента керування, але можуть бути змінені програмістом. Багато значень атрибутів можна змінювати під час виконання на основі дій користувача або змін у середовищі, забезпечуючи динамічну програму. Наприклад, код можна вставити в обробник події зміни розміру форми, щоб змінити положення елемента керування таким чином, щоб він залишався в центрі форми, розширювався, щоб заповнити форму тощо. Вставивши код в обробник події для натискання клавіші в текстовому полі, програма може автоматично перекладати регістр тексту, що вводиться, або навіть забороняти вставляти певні символи.

Покрім забезпечення доступу до стандартних елементів керування ОС, таких як кнопка, текстове поле, прапорець і список, у Windows Forms додано власні елементи керування для розміщення ActiveX, розташування макета, перевірки та зв’язування повних даних. Ці елементи керування відображаються за допомогою GDI+.[9]

## 2.3. Алгоритми роботи з даними в процесах обслуговування клієнтів

Одним із поширених алгоритмів роботи з даними є алгоритм сортування даних, зокрема в межах бази даних. Алгоритм сортування – це алгоритм, який розміщує елементи списку в порядку. Найбільш часто використовуваними порядками є числовий порядок і лексикографічний порядок, за зростанням або за спаданням. Ефективне сортування важливе для оптимізації ефективності інших алгоритмів (таких як алгоритми пошуку та злиття), які вимагають, щоб вхідні дані були у відсортованих списках. Сортування також часто буває корисним для канонізації даних і для створення зрозумілих для людини результатів.

Формально результат будь-якого алгоритму сортування повинен задовольняти двом умовам:

– Вихід здійснюється в монотонному порядку (кожен елемент не менший/більший за попередній елемент, відповідно до необхідного порядку).

– Вихід – це перестановка (перевпорядкування зі збереженням усіх вихідних елементів) вхідних даних.

Для оптимальної ефективності вхідні дані повинні зберігатися в структурі даних, яка дозволяє довільний доступ, а не таку, яка дозволяє лише послідовний доступ.

З самого початку обчислювальної техніки проблема сортування привернула багато досліджень, можливо, через складність її ефективного вирішення, незважаючи на її просте, звичне формулювання. Серед авторів ранніх алгоритмів сортування близько 1951 року була Бетті Холбертон, яка працювала над ENIAC і UNIVAC.[1, 2] Бульбашковий метод був проаналізований ще в 1956 році.[3] Асимптотично оптимальні алгоритми відомі з середини 20 століття – нові алгоритми все ще винаходяться, причому широко використовуваний Тімсорт датується 2002 роком, а бібліотечне сортування вперше було опубліковано в 2006 році.

Алгоритми порівняльного сортування мають фундаментальну вимогу Ω(n log n) порівнянь (деякі вхідні послідовності потребуватимуть кратних n log n порівнянь, де n – кількість елементів у масиві, який потрібно відсортувати). Алгоритми, які не базуються на порівняннях, наприклад сортування підрахунком, можуть мати кращу продуктивність.

Алгоритми сортування поширені на початкових уроках інформатики, де велика кількість алгоритмів для вирішення проблеми забезпечує м’яке ознайомлення з різними основними концепціями алгоритмів, такими як нотація великого O, алгоритми «розділяй і володарюй», структури даних, такі як купи та двійковий код. дерева, рандомізовані алгоритми, аналіз найкращого, найгіршого та середнього випадків, компроміси між часом і простором, верхня та нижня межі.

Оптимальне сортування невеликих масивів (з найменшою кількістю порівнянь і обмінів) або швидке (тобто враховуючи специфічні для машини деталі) все ще є відкритою проблемою дослідження, а рішення відомі лише для дуже маленьких масивів (<20 елементів). Так само оптимальне (за різними визначеннями) сортування на паралельній машині є відкритою темою дослідження.

Найкраща, найгірша та середня поведінка з точки зору розміру списку. Для типових послідовних алгоритмів сортування хороша поведінка – O(n log n), з паралельним сортуванням O(log2 n), а погана – O(n2). Ідеальною поведінкою для послідовного сортування є O(n), але це неможливо в середньому випадку. Оптимальним паралельним сортуванням є O(log n).

Використання пам'яті (і використання інших ресурсів комп'ютера). Зокрема, деякі алгоритми сортування є «на місці». Власне, сортування на місці потребує лише O(1) пам’яті, крім елементів, які сортуються; іноді O(log n) додаткової пам'яті вважається "на місці".

Рекурсія: деякі алгоритми є або рекурсивними, або нерекурсивними, тоді як інші можуть бути обома (наприклад, сортування злиттям).

Стабільність: стабільні алгоритми сортування зберігають відносний порядок записів з однаковими ключами (тобто значеннями).

Незалежно від того, чи є вони сортом для порівняння. Сортування порівняння перевіряє дані лише шляхом порівняння двох елементів за допомогою оператора порівняння.

Загальний метод: вставка, обмін, вибір, об'єднання тощо. Сортування обміном включає спливаюче сортування та швидке сортування. Сортування вибору включає циклічне сортування та сортування по купі.

Чи є алгоритм послідовним чи паралельним. Решта цього обговорення майже виключно зосереджується на послідовних алгоритмах і передбачає послідовну роботу.

Адаптивність: чи впливає попереднє сортування вхідних даних на час роботи. Алгоритми, які враховують це, як відомо, є адаптивними.

Онлайн: такий алгоритм, як Insertion Sort, який є онлайн, може сортувати постійний потік введення.

Стабільні алгоритми сортування сортують однакові елементи в тому самому порядку, в якому вони з'являються у вхідних даних. Наприклад, у прикладі сортування карт праворуч карти сортуються за їхнім рангом, а їхня масть ігнорується. Це дає можливість створювати кілька різних правильно відсортованих версій вихідного списку. Стабільні алгоритми сортування вибирають один із них відповідно до наступного правила: якщо два елементи порівнюються як рівні (наприклад, дві 5 карток), тоді їхній відносний порядок буде збережено, тобто якщо один буде перед іншим у вхідних даних, він прийде перед іншим на виході.

Стабільність важлива для збереження порядку в кількох сортуваннях

одного набору даних. Наприклад, припустимо, що записи студентів, які складаються з імен і класів, сортуються динамічно, спочатку за іменами, а потім за класами. Якщо в обох випадках використовується стабільний алгоритм сортування, операція сортування за класом-розділом не змінить порядок імен; з нестабільним сортуванням може статися, що сортування за розділами перемішує порядок імен, що призводить до неалфавітного списку студентів.

Більш формально, дані, які сортуються, можуть бути представлені як запис або кортеж значень, а частина даних, яка використовується для сортування, називається ключем. У прикладі з картою карти представлені у вигляді запису (рангу, масті), а ключем є ранг. Алгоритм сортування є стабільним, якщо коли є два записи R і S з однаковим ключем і R з'являється перед S у вихідному списку, тоді R завжди з'являтиметься перед S у відсортованому списку.

Коли рівні елементи нерозрізнені, як-от цілі числа, або, загалом, будь-які дані, де весь елемент є ключовим, стабільність не є проблемою. Стабільність також не є проблемою, якщо всі клавіші різні.

Нестабільні алгоритми сортування можуть бути спеціально реалізовані, щоб бути стабільними. Один із способів зробити це – штучно розширити порівняння ключів, щоб порівняння між двома об'єктами з однаковими ключами вирішувалося з використанням порядку записів у вихідному списку вхідних даних як розриву зв'язків. Однак для запам'ятовування цього порядку може знадобитися додатковий час і простір.

## ВИСНОВКИ ДО РОЗДІЛУ 2

У даному розділі проведено кілька ключових етапів у процесі розробки системи. По-перше, була побудована ER-діаграма сутностей, яка послужила основою для подальшого проектування бази даних розроблюваної системи. Цей етап дозволяє чітко визначити взаємозв'язки між різними об'єктами та їх атрибутами.

Далі, був проведений аналіз та вибір технологічного стеку для розробки системи. В результаті вибору платформи .NET, мови програмування C# та середовища Visual Studio було обрано оптимальний набір інструментів для досягнення поставлених цілей. При цьому були визначені конкретні переваги вибраного стеку технологій порівняно з іншими альтернативами.

Крім того, в розділі наведено ряд причин, які обумовили вибір вказаного технологічного стеку. Це може включати в себе ефективність розробки, широкий функціонал платформи .NET, а також велику спільноту розробників.

Окрема увага приділена опису алгоритмів сортування та обробки даних, які були використані у процесі розробки системи. Це важливий аспект, оскільки ефективність обробки даних визначає швидкодію та продуктивність системи.

Загалом, розділ надає повний огляд ключових виборів та етапів, які вплинули на подальший процес розробки системи.

# РОЗДІЛ 3
# ПРОЕКТУВАННЯ І РОЗРОБКА СИСТЕМИ ОБЛУГОВУВАННЯ КЛІЄНТІВ ВЕТЕРИНАРНОЇ КЛІНІКИ

## 3.1 Аналіз варіантів використання системи обслуговування клієнтів

Спочатку необхідно розробити майбутній функціонал системи. Для ефективного визначення системних вимог можна використовувати моделювання поведінки системи за допомогою діаграм варіантів використання в рамках мови моделювання UML. Щоб уточнити функції та межі системи, діаграми варіантів використання надають загальний огляд системи на вищому рівні. Вони також визначають інтерфейс між системою та її акторами. Варіанти використання та актори на діаграмах деталізують дії системи та способи їх використання акторами, але не вдаваються внутрішні деталі роботи системи. Діаграми варіантів використання служать для ілюстрації та визначення контекстуальних та основних вимог для всієї системи або її критичних компонентів. Ці діаграми можна використовувати для моделювання складної системи в цілому або для створення різних діаграм для представлення окремих частин системи. Зазвичай діаграми варіантів використання розробляються на початкових етапах проекту та служать орієнтиром на протязі усього процесу розробки.

Діаграми варіантів використання є важливим інструментом в різних випадках:

1. Початок проекту: Перед початком проекту можна ефективно створити діаграми варіантів використання для моделювання бізнесу. Це допомагає всім учасникам проекту спільно розібратися в ролях працівників, клієнтів і ключових процесах бізнесу.

2. Збір вимог: Під час збору вимог діаграми варіантів використання служать ефективним інструментом для відображення системних вимог. Вони дозволяють

чітко представити завдання системи та сприяють кращому розумінню вимог серед учасників проекту.

3. Аналіз та проектування: На етапах аналізу та проектування діаграми варіантів використання допомагають визначити класи, які необхідні для системи, використовуючи інформацію про варіанти використання та акторів.

4. Тестування: Під час етапу тестування діаграми варіантів використання використовуються для визначення тестових сценаріїв та випробувань системи, сприяючи вдосконаленню якості програмного продукту.

Варіант використання представляє операцію, яку система виконує для досягнення конкретної мети користувача, при цьому забезпечуючи видимий результат, що має цінність для користувача системи.

У межах UML підсистеми виступають як тип стереотипних компонентів. Вони є представленням незалежних поведінкових одиниць у системі. Використання підсистем демонструється в діаграмах класів, компонентів та варіантів використання, щоб лаконічно відобразити великомасштабні компоненти системи, які ви моделюєте.

В рамках UML термін "зв'язок" означає взаємозв'язок між різними елементами моделі. Відношення UML є концепцією, яка додає семантику до моделі, чітко визначаючи структуру та взаємодію між різними елементами моделі.

На діаграмі варіантів використання UML ці відносини можуть бути графічно зображені та ілюстровані у вигляді рисунка 3.1.

Діаграма варіантів використання розширюється для відображення всіх потенційних дій користувача та їх взаємозв'язку з системою на конкретному етапі.

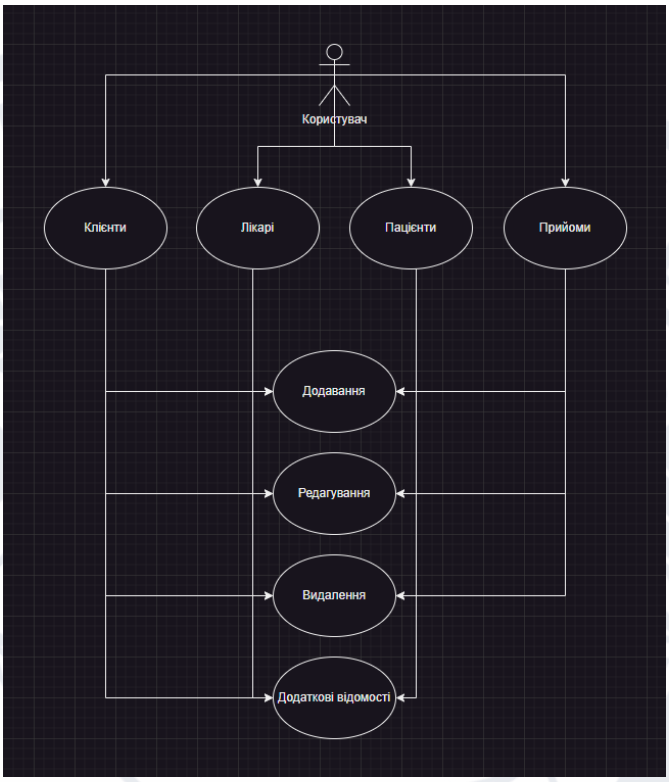


Рис. 3.1 – Діаграма способів використання додатку

## 3.2 Проектування внутрішньої структури системи

Для проектування внутрішньої структури системи було розроблено UML діаграму класів. В рамках UML діаграми класів визначаються як один із шести типів структурних діаграм. Ці діаграми становлять основу для процесу моделювання об'єктів і призначені для відображення статичної структури системи. Залежно від складності системи можна використовувати одну діаграму класів для моделювання всієї системи або використовувати кілька діаграм для конкретних компонентів системи.

Визначаючи структуру вашої системи або підсистеми, діаграми класів стають своєрідним картафельним планом. За їх допомогою ви можете моделювати об'єкти, що компонують систему, відобразити зв'язки між ними та докладно описати функції та сервіси, які ці об'єкти здійснюють.

Діаграми класів є важливим інструментом на різних етапах розробки системи. На етапі аналізу вони сприяють розумінню вимог для конкретної області та визначенню компонентів системи. Під час розробки об'єктно-орієнтованого програмного забезпечення діаграми класів, розроблені на початкових етапах, часто трансформуються у фактичні класи та об'єкти програмного забезпечення під час кодування. Пізніше ці діаграми можуть бути вдосконалені, відображаючи конкретні аспекти системи, такі як інтерфейси користувача, логічні реалізації та інше. Таким чином, діаграми класів стають інструментом, який дозволяє чітко уявити та зрозуміти внутрішню логіку розроблюваної системи, надаючи зрозуміле відображення зв'язків між її компонентами на різних рівнях.

Ви можете використовувати діаграми класів для візуалізації, визначення та документування структурних аспектів у своїх моделях. Наприклад, на етапах аналізу та проектування під час циклу розробки, ви можете створювати діаграми класів для виконання таких завдань:

1. Зберіть і визначте структуру класів та інших класифікаторів.
2. Визначте зв’язки між класами та класифікаторами.
3. Проілюструйте структуру моделі за допомогою атрибутів, операцій і сигналів.
4. Покажіть загальні ролі та обов’язки класифікатора, які визначають поведінку системи.
5. Показати класи реалізації в пакеті..
6. Покажіть структуру та поведінку одного чи кількох класів
7. Показати ієрархію успадкування серед класів і класифікаторів.
8. Покажіть працівників і сутності як моделі бізнес-об’єктів.

На етапі впровадження циклу розробки програмного забезпечення, використання діаграм класів дозволяє перетворити ваші моделі у код і, навпаки,

перетворити написаний код в моделі. У мові моделювання UML, клас представляє об'єкт або групу об'єктів, які об'єднуються спільною структурою та поведінкою. Класи або їх екземпляри є основними елементами моделей, які можна відобразити на діаграмах UML.

У контексті UML об'єкти є елементами моделі, які репрезентують екземпляри класів чи самі класи. Додавання об'єктів до моделі дозволяє вам конкретизувати представлення конкретних та прототипних екземплярів. Конкретний екземпляр відображає реальний об'єкт або особу у реальному світі. Наприклад, конкретний екземпляр класу "Клієнт" може представляти конкретного клієнта. Прототипний екземпляр класу "Клієнт" містить дані, які представляють типового клієнта.

У контексті UML, сигнали є незалежними елементами моделі, не пов'язаними з класифікаторами, які обробляють їх. Сигнали визначають односторонній асинхронний зв'язок між активними об'єктами. Перерахування в UML виступають як елементи моделі в діаграмах класів, що представляють типи даних, визначені користувачем. Вони містять набори іменованих ідентифікаторів, що представляють значення перерахування, відомі як літерали перерахування.

Додатково, типи даних в UML – це елементи моделі, які визначають значення даних. Зазвичай вони використовуються для представлення примітивних типів, таких як цілі або рядкові типи, і також для перерахувань, що визначені користувачем.

У UML-моделях артефакти представляють фізичні сутності у програмній системі. Вони є фізичними одиницями реалізації, такими як виконувані файли, бібліотеки, програмні компоненти, документи та бази даних. У контексті UML, зв'язки є взаємозв'язками між елементами моделі, а відношення UML – це тип елемента моделі, який призначений додати семантику до моделі, конкретизуючи

структуру та поведінку між елементами. Кваліфікатори в UML виступають як властивості бінарних асоціацій і можуть бути частиною кінців асоціації. Вони включають список атрибутів асоціації, кожен з яких має назву та тип. Ці атрибути асоціації моделюють ключі, використовувані для індексації підмножини екземплярів зв'язку.

На рисунку 3.2 зображено діаграму класів системи.


Рис. 3.2 – Діаграма класів системи

## 3.3 Розробка графічного інтерфейсу користувача системи обслуговування клієнтів ветеринарної клініки

Графічний інтерфейс користувача (GUI) – це комплекс взаємодії з користувачем у комп'ютерному програмному забезпеченні за допомогою візуальних компонентів. GUI відображає об'єкти, які несуть інформацію та представляють доступні дії для користувача. Взаємодія з цими об'єктами може змінювати їх колір, розмір або видимість. Графічний інтерфейс користувача був розроблений вперше у 1981 році в Xerox PARC Аланом Кеєм, Дугласом Енгельбартом та їхньою дослідницькою групою. Пізніше, 19 січня 1983 року,

Apple представила комп'ютер Lisa, який використовував графічний інтерфейс користувача.

Графічний інтерфейс включає в себе різноманітні графічні елементи, такі як значки, курсори та кнопки, які полегшують взаємодію користувача з комп'ютерною системою. Ці графічні компоненти можуть бути вдосконалені за допомогою звукових ефектів або візуальних особливостей, таких як прозорість і тіні. Використовуючи ці об'єкти, користувач може взаємодіяти з комп'ютером, не потрібно знати складні команди.

Нижче представлене зображення робочого столу операційної системи Windows 7, а також ілюстрація використання графічного інтерфейсу користувача. У цьому прикладі можна використовувати мишу для переміщення вказівника та клацання на піктограму програми для її запуску.

Щоб забезпечити максимальний комфорт для користувача у графічному інтерфейсі (GUI), використовуються різноманітні елементи та об'єкти для спрощення взаємодії з програмним забезпеченням. Нижче подано список цих елементів разом з коротким описом кожного з них:

1. Значки (Icons): Мініатюрні графічні зображення, які представляють програми, файли або функції. Користувач може клікнути на значок для виконання певної операції.

2. Кнопки (Buttons): Елементи інтерфейсу, які використовуються для запуску або виконання конкретної дії. Клацання по кнопці викликає відповідну функцію чи команду.

3. Меню (Menus): Відкриваються списки команд чи опцій, які користувач може вибрати. Меню розгортаються при натисканні або наведенні на відповідний об'єкт.

4. Вікна (Windows): Графічні області, що відображають вміст програми чи системи. Можуть бути переміщені, змінені розміром та закриті.

5. Текстові поля (Text Fields): Елементи для введення текстової інформації. Використовуються для введення даних, таких як імена файлів чи текстові повідомлення.

6. Полоси прокрутки (Scroll Bars): Елементи інтерфейсу, що дозволяють користувачеві прокручувати вміст, який не вміщується на екрані.

7. Панелі інструментів (Toolbars): Стрічки іконок або кнопок, які забезпечують швидкий доступ до найчастіше використовуваних функцій.

8. Діалогові вікна (Dialog Boxes): Вікна, що виходять на передній план і вимагають від користувача введення або вибору опцій перед продовженням роботи.

9. Чекбокси та перемикачі (Checkboxes and Switches): Елементи для вибору опцій чи включення/виключення певних параметрів.

10. Покажчики (Pointers): Символи або вказівники, які вказують на поточне місцезнаходження користувача чи виділяють важливі області на екрані.

Забезпечення доступу до цих елементів робить взаємодію з програмним забезпеченням інтуїтивно зрозумілою та зручною для користувача.

Графічний інтерфейс використовує вікна, піктограми та меню для виконання різних команд, таких як відкриття, видалення та переміщення файлів. В основному, навігація в операційній системі з графічним інтерфейсом здійснюється за допомогою миші; проте клавіатуру також можна використовувати, використовуючи комбінації клавіш або стрілкові клавіші.

Наприклад, для відкриття програми в операційній системі з графічним інтерфейсом користувач може навести вказівник миші на піктограму програми та двічі клікнути по ній. У випадку використання інтерфейсу командного рядка, користувач повинен знати конкретні команди, такі як перехід до каталогу, що містить програму, виведення списку файлів і запуск конкретного файлу.

Графічний інтерфейс користувача розглядається як більш зручний у

порівнянні з текстовим інтерфейсом командного рядка, таким як MS-DOS або оболонка Unix-подібних операційних систем. Відмінності від операційних систем з командним рядком (CUI), таких як Unix або MS-DOS, полягають в тому, що операційні системи з графічним інтерфейсом користувача набагато легше вивчати та використовувати, оскільки не потрібно запам'ятовувати команди. Крім того, користувачам не потрібно володіти мовами програмування. Ця простота використання та більш сучасний зовнішній вигляд дозволили операційним системам з графічним інтерфейсом користувача здобути домінування на сучасному ринку. Пристрій для вказівки, такий як миша, є ключовим елементом взаємодії з майже всіма аспектами графічного інтерфейсу користувача (GUI). В сучасному оточенні також використовуються сенсорні екрани для введення команд та взаємодії з пристроями.

На рис. 3.3 показана сторінка зі створення нових лікарів, клієнтів, пацієнтів(тварин) та прийомів в режимі створення і додавання нових сутностей для наповнення бази даних тестовими даними.

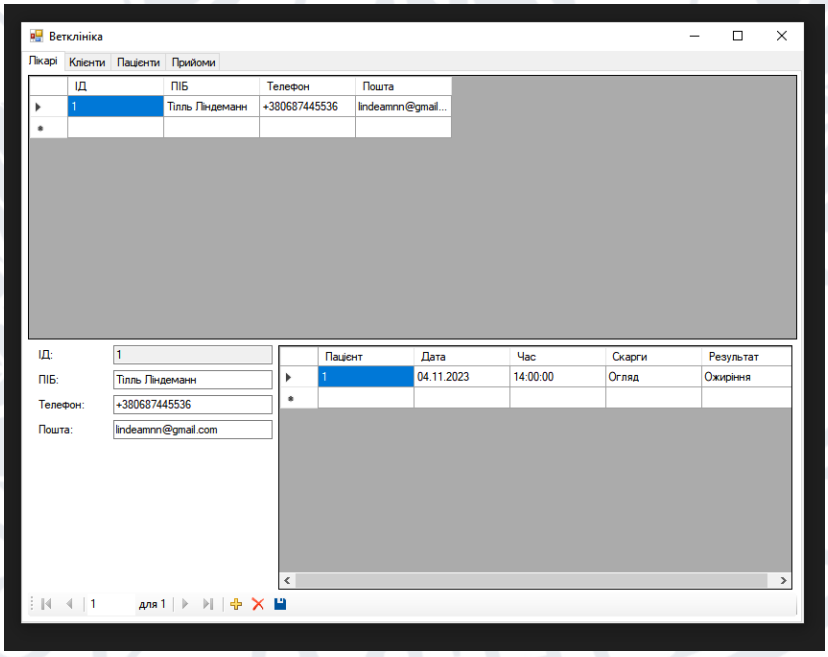


Рис. 3.3 – Сторінка лікарів

На рисунку 3.4 показана сторінка уведення та аналізу даних по проботі з клієнтами.

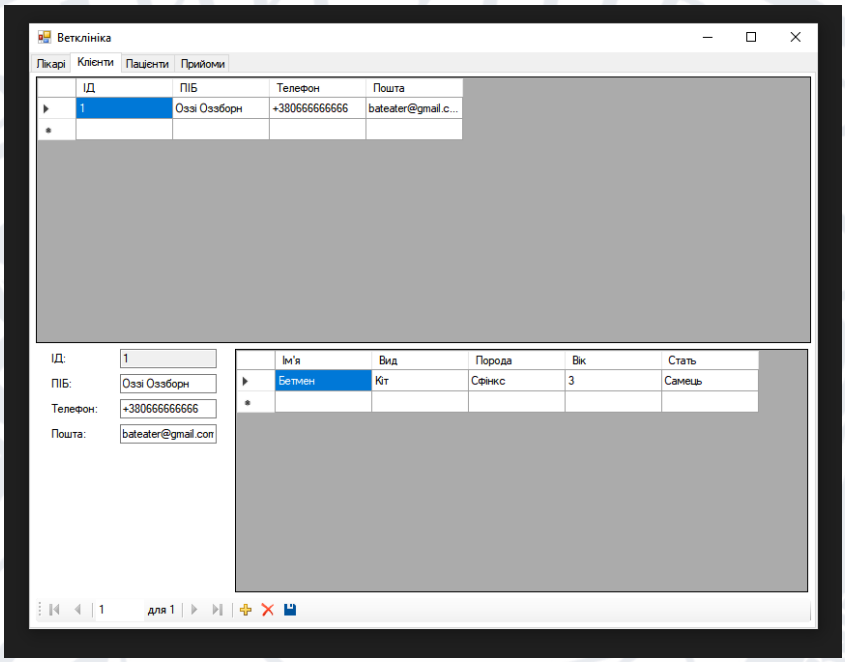


Рис. 3.4 – Сторінка клієнтів

На рисунку 3.5 наведено сторінку для уведення та обробки даних пацієнтів (тварин).

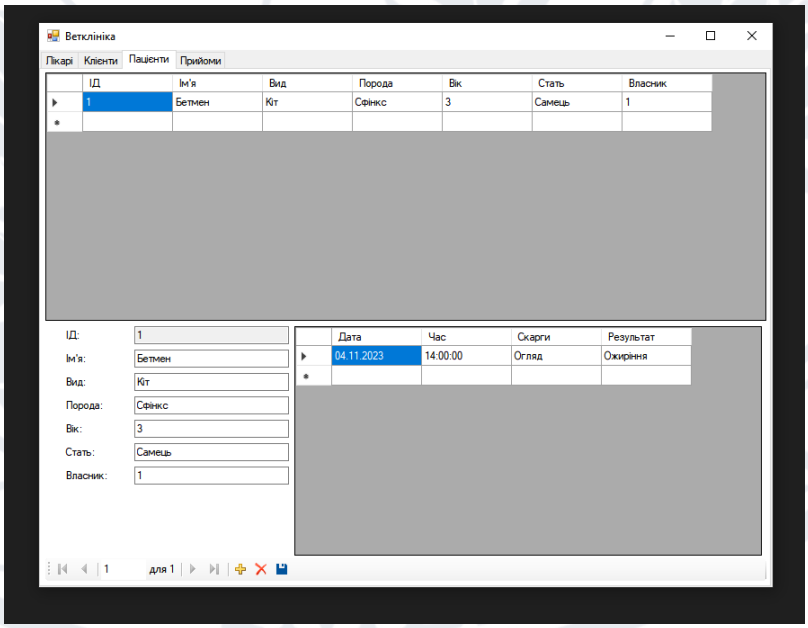


Рис. 3.5 – Сторінка пацієнтів

Сторінка прийомів ведеться за формою показаною на рисунку 3.6.

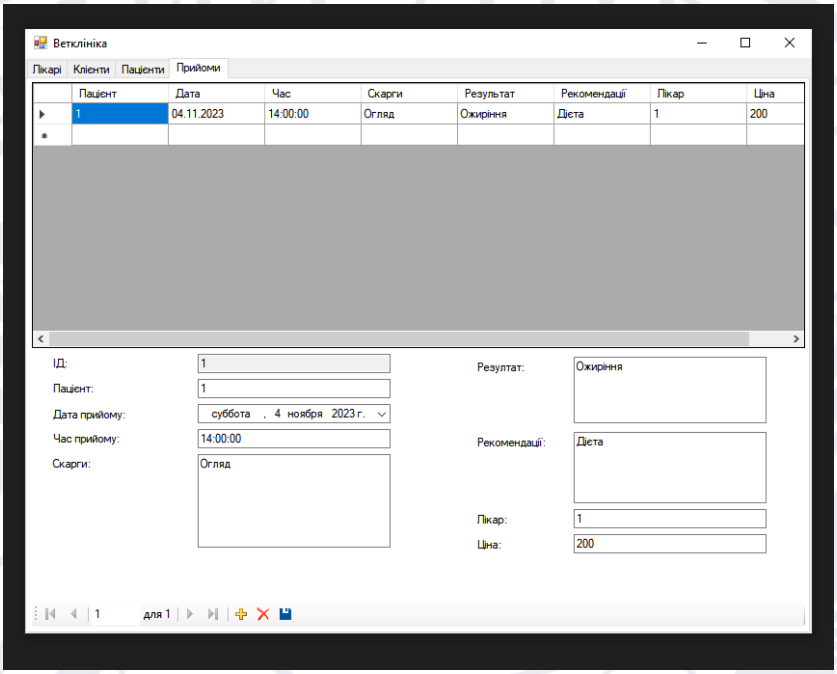


Рис. 3.6 – Сторінка прийомів

Таким чином, основними елементами для введення, роботи та аналізу даних є сторінки: лікарів, пацієнтів та прийомів.

## ВИСНОВКИ ДО РОЗДІЛУ 3

Створений інтерфейс програми має простоту та зручну систему взаємодії для користувача. Його інтуїтивний дизайн дозволяє легко здійсювати навігацію та використовувати всі доступні функції без великих зусиль. Графічне оформлення та розміщення елементів інтерфейсу відповідають сучасним стандартам, що робить використання програми приємним для користувача.

Щодо функціоналу, програма надає широкий спектр можливостей, які відповідають потребам користувачів. Відзначається особливою ефективністю в роботі з базою даних, що дозволяє швидко та зручно зберігати, оновлювати та вивчати інформацію. Передбачено не лише базовий функціонал для керування клієнтською інформацією, але й додаткові можливості, які сприяють полегшенню робочого процесу ветеринарних фахівців.

Загалом, комбінація інтуїтивно зрозумілого інтерфейсу та розширеного функціоналу робить цю програму ефективним інструментом для ветеринарних клінік, забезпечуючи високий рівень зручності та продуктивності.

# ВИСНОВКИ

За результатами дослідження методів та інформаційних технологій обробки даних процесів обслуговування клієнтів ветеринарних клінік були зроблені наступні висновки:

1. Актуальність використання сучасних інформаційних технологій та методів аналізу даних процесів обслуговування клієнтів ветеринарних клінік обумовлена такими причинами, як зростаючий рівень інформатизації бізнес-процесів, соціальні чинники, необхідність актуалізації та достовірності даних, забезпечення доступності та якості обробки даних.

2. Ідеологія ведення обслуговування клієнтів засобами взаємодії лікарів-пацієнтів через використання системної інформатизації процесів обгрунтовується теоретичними положеннями побудови інформаційних систем, що відтворюють концепцію їх використання з урахуванням особливостей ветеринарних клінік.

3. Проведений аналіз існуючих додатків/веб-сайтів ветеринарних клінік, функціональне призначення яких є подібним до проектного рішення системи, запропонованого в даній роботі, дозволило виявити основні позитивні риси розробок, серед яких можна вважати: необхідність відтворення зручного користувацького інтерфейсу, забезпечення високого рівня доступності інформації, реалізація онлайн-записів до лікаря, можливість отримання онлайн-консультації.

4. Проведений аналіз існуючих інструментальних засобів для розробки системи аналізу даних процесів обслуговування клієнтів ветеринарної клініки дозволив обґрунтувати використання об’єктно-орієнтованого програмування для розробки системи з додатковими можливостями запису та реєстрації.

5. Розроблений програмний продукт реалізує конпецію інформаційної системи орієнтована на концептуальні положення CRM-систем ефективно

вирішує проблеми централізованого підходу до обробки даних процесів обслуговування клієнтів, забезпечує підвищення якості обслуговування та раціональне використання ресурсів.

6. Повний цикл обслуговування клієнтів ветеринарної клініки в спроектованій системі досягається шляхом інтеграції ключових модулів (реєстрація клієнтів та їхніх тварин, запис на прийом, електронна картка пацієнта, управління запасами).

7. Авторський проект бази даних реалізує взаємодію категорій об’єктно-орієнтованого програмування через реалізацію сутностей: лікар, користувач (клієнт), пацієнт (тварина), прийом, ліки, діагноз, послуги.

8. Спроектовано та здійснено практичну реалізацію користувацького інтерфейсу системи обслуговування клієнтів ветеринарної клініки, який відтворює можливі функціональні особливості варіантів використання.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ І ЛІТЕРАТУРИ

1. InfoQ eMag: A Preview of C# 7. URL: https://www.infoq.com/minibooks/emag-c-sharp-preview/

2. Welcome to C# 11. Retrieved December 10, 2022. URL: https://devblogs.microsoft.com/dotnet/welcome-to-csharp-11/

3. Torgersen, Mads (2008). New features in C# 4.0. Microsoft. URL: https://www.c-sharpcorner.com/uploadfile/john_charles/new-features-of-C-Sharp-4-0/

4. The Roslyn .NET compiler provides C# and Visual Basic languages with rich code analysis APIs.: dotnet/roslyn". November 13, 2019. URL: https://learn.microsoft.com/uk-ua/dotnet/csharp/whats-new/breaking-changes .

5. CoreCLR is the runtime for .NET Core. It includes the garbage collector, JIT compiler, primitive data types and low-level classes.: dotnet/coreclr. November 13, 2019. URL: https://github.com/dotnet/coreclr

6. Naugler David (2007). C# 2.0 for C++ and Java programmer: conference workshop. Journal of Computing Sciences in Colleges. 22 (5). URL: https://dl.acm.org/doi/10.5555/1229688.1229689

7. Hamilton, Naomi (2008). The A-Z of Programming Languages: C#. Computerworld. URL: https://www2.computerworld.com.au/article/261958/a-z_programming_languages_c_/

8. Chapel spec (Acknowledgments). Cray Inc. 2015. URL: https://chapel-lang.org/docs/language/spec/acknowledgements.html

9. Rich Hickey Q&A by Michael Fogus. Archived from the original on January 11, 2017. URL: https://gist.github.com/athos/c880034578b7bc5b3603

10. Borenszweig Ary (2016). Crystal 0.18.0 released! URL: https://crystal-lang.org/2016/06/14/crystal-0.18.0-released/

11. Web Languages and VMs: Fast Code is Always in Fashion. (V8, Dart) -

Google I/O 2013". YouTube. Archived from the original on December 21, 2021. URL: https://www.youtube.com/watch?v=huawCRlo9H4

12. Java 5.0 added several new language features (the enhanced for loop, autoboxing, varargs and annotations), after they were introduced in the similar (and competing) C# language . URL: https://www3.ntu.edu.sg/home/ehchua/programming/java/JDK5_NewFeatures.html

13. Cornelius, Barry (December 1, 2005). "Java 5 catches up with C#". University of Oxford Computing Services. URL: https://softwareengineering.stackexchange.com/questions/63964/will-java-catch-up-with-c

14. Influences - The Rust Reference. The Rust Reference. URL: https://doc.rust-lang.org/reference/influences.html

15. Lattner Chris (2014). Chris Lattner's Homepage. Chris Lattner. URL: https://www.nondot.org/sabre/

16. Skeet 2019.

17. C# Language Specification (PDF) (4th ed.). Ecma International. June 2006. URL: https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/introduction

18. Torgersen, Mads (November 8, 2022). "Welcome to C# 11". URL: https://devblogs.microsoft.com/dotnet/welcome-to-csharp-11/

19. Announcing .NET 7 – the Fastest .NET Yet. November 11, 2022.[permanent dead link]. URL: https://devblogs.microsoft.com/dotnet/announcing-dotnet-7/

20. Albahari 2022. https://www.researchgate.net/publication/361115135_9_Albahari_et_al_2022 .

21. Design Goals of C#. www.java-samples.com. URL: https://www.java-samples.com/ .

22. Zander, Jason (2007). Couple of Historical Facts. URL: https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

23. What language was ASP.Net originally written in?. November 28, 2006. Archived from the original on June 24, 2016. URL: https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet

24. Hamilton, Naomi (2008). The A-Z of Programming Languages: C#. Computerworld. Archived from the original on May 18, 2019. URL: https://www2.computerworld.com.au/article/261958/a-_programming_languages_c_/

25. Details. nilsnaegele.com. Archived from the original on April 7, 2019.

26. Why Microsoft's C# isn't. CNET: CBS Interactive. 2002. URL: https://www.cnet.com/tech/tech-industry/why-microsofts-c-isnt/

27. Bill Joy (2002). Microsoft's blind spot. cnet.com. URL: https://www.cnet.com/tech/tech-industry/microsofts-blind-spot/

28. Osborn, John (2000). Deep Inside C#: An Interview with Microsoft Chief Architect Anders Hejlsberg. O'Reilly Media. URL: https://www.codebrary.com/2018/03/deep-inside-c-sharp-interview-with.html

29. Generics (C# Programming Guide). Microsoft. URL: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/generic-classes

30. Don Box and Anders Hejlsberg (February 2007). LINQ: .NET Language-Integrated Query. Microsoft. URL: https://medium.com/knowledge-pills/what-is-linq-in-c-ffec88ee7261

31. Mercer, Ian (2010). Why functional programming and LINQ is often better than procedural code. abodit.com. Archived from the original on July 11, 2011. URL: https://learn.microsoft.com/en-us/dotnet/standard/linq/functional-vs-imperative-programming

32. Andy Retires. Dan Fernandez's Blog. Blogs.msdn.com. January 29, 2004.

Archived from the original on January 19, 2016. URL: https://learn.microsoft.com/en-us/archive/blogs/ .

33. Technical committees - JTC 1/SC 22 - Programming languages, their environments and system software interfaces". ISO. URL: https://www.iso.org/committee/45202.html .

34. ISO/IEC 23270:2003 - Information technology - C# Language Specification. Iso.org. August 23, 2006. Archived from the original on May 8, 2012. URL: https://www.iso.org/committee/45202.html

35. ISO/IEC 23270:2006 - Information technology - Programming languages - C#. Iso.org. January 26, 2012. URL: https://www.iso.org/committee/45202.html

36. ISO/IEC 23270:2018 Information technology – Programming languages – C#. ISO. URL: https://www.iso.org/committee/45202.html

37. Mariani, Rico (2009). My History of Visual Studio (Part 1) – Rico Mariani's Performance Tidbits. Rico Mariani's Performance Tidbits. URL: https://ricomariani.medium.com/my-history-of-visual-studio-5652e3e74af

38. Kovacs, James (2007). C#/.NET History Lesson". URL: https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history

39. Hejlsberg, Anders (October 1, 2008). The A-Z of Programming Languages: C#. Computerworld. Archived from the original on April 2, 2015. URL: https://www2.computerworld.com.au/article/261958/a-z_programming_languages_c_/

40. Microsoft C# FAQ. Microsoft. Archived from the original on February 14, 2006. URL: https://learn.microsoft.com/uk-ua/

41. F# FAQ. Microsoft Research. Archived from the original on February 18, 2009. URL: https://learn.microsoft.com/uk-ua/

42. Simon, Raphael; Stapf, Emmanuel; Meyer, Bertrand (2002). Full Eiffel on the .NET Framework. Microsoft. URL: https://devblogs.microsoft.com/

43. What's new in the C# 2.0 Language and Compiler. Microsoft. Archived from the original on December 18, 2010. URL: https://devblogs.microsoft.com/

44. Hejlsberg, Anders; Torgersen, Mads (2007). Overview of C# 3.0. Microsoft Developer Network. Microsoft. URL: https://devblogs.microsoft.com/ .

45. Using C# 3.0 from .NET 2.0. Danielmoth.com. 2007. URL: http://www.danielmoth.com/Blog/using-c-30-from-net-20.aspx .

46. Hejlsberg, Anders. Future directions for C# and Visual Basic. C# lead architect. Microsoft. 2011.

47. An Introduction to New Features in C# 5.0". MSDN Blogs. Microsoft. 2014. URL: https://devblogs.microsoft.com/

48. Language feature implementation status. github. Microsoft. 2016. URL: https://learn.microsoft.com/uk-ua/.

49. What's new in C# 7. Microsoft Docs. 2016. URL: https://learn.microsoft.com/uk-ua/.

50. New Features in C# 7.0. .NET Blog. 2017. URL: https://devblogs.microsoft.com/

51. Visual Studio 2017 version 15.0 Release Notes. docs.microsoft.com. 2023. URL: https://learn.microsoft.com/uk-ua/

52. What's new in C# 7.1. Microsoft Docs. URL: https://learn.microsoft.com/uk-ua/

53. Visual Studio 2017 version 15.3 Release Notes. docs.microsoft.com. April 11, 2023. URL: https://learn.microsoft.com/uk-ua/

54. What's new in C# 7.2. Microsoft Docs. URL: https://learn.microsoft.com/uk-ua/

55. Visual Studio 2017 version 15.5 Release Notes. docs.microsoft.com. April 11, 2023. URL: https://learn.microsoft.com/uk-ua/ .

56. What's new in C# 7.3. Microsoft Docs. URL:

https://learn.microsoft.com/uk-ua/

57. Visual Studio 2017 version 15.7 Release Notes. docs.microsoft.com. July 13, 2022. URL: https://learn.microsoft.com/uk-ua/

58. What's new in C# 8.0. Microsoft Docs. 2023. URL: https://learn.microsoft.com/uk-ua/ .

59. Visual Studio 2019 version 16.3 Release Notes. docs.microsoft.com. April 11, 2023. URL: https://learn.microsoft.com/uk-ua/

60. Bill Wagner. What's new in C# 9.0 - C# Guide. docs.microsoft.com. URL: https://learn.microsoft.com/uk-ua/

61. Visual Studio 2019 version 16.8 Release Notes. docs.microsoft.com. 2023. URL: https://learn.microsoft.com/uk-ua/

62. What's new in C# 10. docs.microsoft.com. URL: https://learn.microsoft.com/uk-ua/

63. Visual Studio 2022 version 17.0 Release Notes. docs.microsoft.com. URL: https://learn.microsoft.com/uk-ua/

64. What's new in C# 11. docs.microsoft.com. URL: https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-11

65. Visual Studio 2022 version 17.4 Release Notes. docs.microsoft.com. URL: https://learn.microsoft.com/en-us/visualstudio/releases/2022/release-notes-v17.4

66. What's new in C# 12. docs.microsoft.com. URL: https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-12 .

67. Visual Studio 2022 version 17.6 Release Notes. docs.microsoft.com. URL: https://learn.microsoft.com/en-us/visualstudio/releases/2022/release-notes-v17.6

68. Novák et al. 2010.

69. Steve Whims, Matt Wojo (2022). Compiling Apps with .NET Native - UWP applications. learn.microsoft.com. URL: https://learn.microsoft.com/uk-ua/

70. LakshanF; agocke; Rick Anderson; et al. (2023). Native AOT deployment

overview - .NET". learn.microsoft.com. URL: https://learn.microsoft.com/uk-ua/ .

71. BillWagner. Expression Trees (C#)". docs.microsoft.com. URL: https://learn.microsoft.com/en-us/dotnet/csharp/advanced-topics/expression-trees/ .

72. Dotnet-bot. System.Reflection.Emit Namespace. learn.microsoft.com. URL: https://learn.microsoft.com/uk-ua/

73. McAllister, Neil (2011). Microsoft's Roslyn: Reinventing the compiler as we know it. InfoWorld. URL: https://www.infoworld.com/article/2078453/microsoft-s-roslyn--reinventing-the-compiler-as-we-know-it.html

74. Introducing C# Source Generators.NET Blog. 2020. URL: https://devblogs.microsoft.com/dotnet/introducing-c-source-generators/

75. Virtual (C# Reference). docs.microsoft.com. 2021. URL: https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/virtual

76. Auto-Implemented Properties (C# Programming Guide). URL: https://learn.microsoft.com/uk-ua/dotnet/csharp/programming-guide/classes-and-structs/auto-implemented-properties

77. Using directive - C# Reference. Microsoft Docs. URL: https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/using-directive

78. BillWagner. Unsafe code, pointers to data, and function pointers. docs.microsoft.com. URL: https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/unsafe-code

79. How to create user-defined exceptions. URL: https://www.geeksforgeeks.org/user-defined-custom-exception-in-java/

80. Venners, Bill; Eckel, Bruce (2003). The Trouble with Checked Exceptions. URL: https://www.artima.com/articles/the-trouble-with-checked-exceptions

81. BillWagner. Operator overloading - C# reference. docs.microsoft.com. URL: https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators

/operator-overloading

82. Zhang, Xue Dong; Teng, Zi Mu; Zhao, Dong Wang (September 2014). "Research of the Database Access Technology Under.NET Framework". Applied Mechanics and Materials. 644–650: 3077–3080. doi:10.4028/www.scientific.net/AMM.644-650.3077. URL: https://www.scientific.net/AMM.644-650.3077

83. Otey, Michael (2006). LINQ to the Future". SQL Server Magazine. Vol. 8, no. 2. pp. 17–21. URL: https://www.itprotoday.com/microsoft-visual-studio/linq-future

84. Sheldon, William (2010). New Features in LINQ". SQL Server Magazine. Vol. 12, no. 11. pp. 37–40. ProQuest 770609095. URL: https://www.ifourtechnolab.com/blog/net-core-6-improvements-in-linq-an-essential-guide-for-2023

85. Bill Wagner (2021). Query Syntax and Method Syntax in LINQ (C#). learn.microsoft.com. URL: https://learn.microsoft.com/en-us/dotnet/csharp/linq/get-started/write-linq-queries

86. Erik Dietrich (2023). The history of C# - C# Guide. learn.microsoft.com. URL: https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history

87. The functional journey of C# - Mads Torgersen - NDC Copenhagen 2022, URL: https://www.infoq.com/presentations/c-sharp-functional-features/

88. The Beauty of Closures. csharpindepth.com. 2023. URL: https://csharpindepth.com/

89. Bill Wagner. Anonymous functions - C# Programming Guide. docs.microsoft.com. 2021. URL: https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/lambda-expressions

90. What's New in C# 7.0. Microsoft Docs. 2019. URL:

https://devblogs.microsoft.com/dotnet/whats-new-in-csharp-7-0/ .

91. C# 9.0 on the record.NET Blog. November 10, 2020. URL: https://devblogs.microsoft.com/dotnet/c-9-0-on-the-record/

92. Bill Wagner (2022). Init keyword - C# Reference. learn.microsoft.com. URL: https://learn.microsoft.com/uk-ua/

93. The .NET Compiler Platform, .NET Platform, 2023, URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/roslyn-sdk/

94. Archer 2001.

95. Lippert, Eric (2009). Representation and Identity. Fabulous Adventures In Coding. Blogs.msdn.com. URL: https://learn.microsoft.com/en-us/archive/blogs/

96. Framework Libraries. docs.microsoft.com. 2023. URL: https://learn.microsoft.com/en-us/dotnet/standard/class-libraries

97. Bill Wagner. What's new in C# 9.0 - C# Guide. docs.microsoft.com. URL: https://learn.microsoft.com/uk-ua/

98. Bill Wagner. Main() and command-line arguments. docs.microsoft.com. URL: https://learn.microsoft.com/uk-ua/.

99. Patent Pledge for Open Source Developers. 2023. URL: https://learn.microsoft.com/en-us/openspecs/dev_center/ms-devcentlp/d45911ba-6532-4d5e-8bdb-719c05172973

100. Patent Cooperation Agreement - Microsoft & Novell Interoperability Collaboration. Microsoft. November 2, 2006.

101. Definitions. Microsoft. November 2, 2006. URL: https://learn.microsoft.com/uk-ua/

102. Steinman, Justin (2006). Novell Answers Questions from the Community. Retrieved July 5, 2009. URL: http://br-linux.org/linux/novell-community-interview

103. Covenant to Downstream Recipients of Moonlight - Microsoft & Novell Interoperability Collaboration. Microsoft. September 28, 2007. URL:

https://www.microsoft.com/en-us/legal/intellectualproperty/tech-licensing/customer-agreements

104. The RyuJIT transition is complete. microsoft.com. June 19, 2018. URL: https://devblogs.microsoft.com/dotnet/the-ryujit-transition-is-complete/

105. Managed Execution Process. microsoft.com. Archived from the original on December 23, 2017. URL: https://www.microsoft.com/uk-ua/

106. Coreclr/src/jit/. github.com. Archived from the original on January 9, 2019.

107. C# Guide. docs.microsoft.com. URL: https://learn.microsoft.com/uk-ua/

108. 5.0.8. microsoft.com. Archived from the original on April 23, 2020. URL: ". https://learn.microsoft.com/uk-ua/

109. Mitigation: New 64-bit JIT Compiler. microsoft.com. 2018. URL: https://www.microsoft.com/uk-ua/

110. Etcheverry, Ignacio (2017). Introducing C# in Godot. Godot Engine. 2018. URL: https://godotengine.org/article/introducing-csharp-godot/

Додаток А

**ДЕКЛАРАЦІЯ**

про дотримання академічної доброчесності

Я, _____

_____

*Повністю вказується ПІБ та статус (посада для працівників, освітня (освітньо-наукова) програма – для здобувачів вищої освіти)*

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

**ЗОБОВ'ЯЗУЮСЬ:**

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

**ПІДТВЕРДЖУЮ:**

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

**УСВІДОМЛЮЮ:**

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)