МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ПрАТ «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Предметно-циклова комісія інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА
Голова ПЦК _____
спеціаліст в/к Сабанов С.О.

ВИПУСКНА РОБОТА МОЛОДШОГО СПЕЦІАЛІСТА

Тема «Розробка прикладної програми для ветеринарної лікарні «Cat-Dog»

Виконав
ст. гр.ІПЗ -118К9 _____ І.В. Агапов

Керівник
викладач _____ Д.О. Костерной

Запоріжжя
2022

ПРАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Предметно-циклова комісія інформаційних технологій

ЗАТВЕРДЖУЮ

Голова ПЦК спеціаліст
в/к Сабанов С.О._____
“ 17 ” січня 2022 року

ЗАВДАННЯ

ВИПУСКНОЇ РОБОТИ МОЛОДШОГО СПЕЦІАЛІСТА

Студента гр. *ІПЗ-118К9*

Спеціальності: *121 – Інженерія програмного забезпечення*

Агапову Ігору Васильовичу

(прізвище, ім'я, по батькові)

1. Тема: «*Розробка прикладної програми для ветеринарної лікарні «Cat-Dog»*

затверджена наказом по інституту: № _____ від 04 березня 2022 року

2. Термін здачі студентом закінченої роботи: 18 червня 2022 року

3. Перелік питань, що підлягають розробці:

1. *Розглянути питання актуальності розробки додатку*

2. *Провести огляд галузі та аналітику проблеми і її рішень загалом;*

3. *Провести огляд та аналіз сучасних аналогів додатків;*

4. *Провести огляд стеку технологій та для розробки проекту та зробити вибір на підставі вимог;*

5. *Виконати всі поставленні задачі випускної роботи;*

6. *Проаналізувати отримані результати;*

7. *Оформити результати у вигляді пояснювальної записки до відповідних ДСТУ норм випускних робіт молодшого спеціаліста.*

Дата видачі завдання: 17 січня 2022 року

4. Календарний графік підготовки випускної роботи молодшого спеціаліста

№ етапу	Зміст	Терміни виконання	Готовність по графіку (%), підпис керівника	Підпис керівника про повну готовність етапу, дата
1	Формулювання (корегування) теми випускної роботи молодшого спеціаліста та збір практичного матеріалу за темою випускної роботи	17.01.22-26.02.22		
2	І атестація І розділ випускної роботи молодшого спеціаліста	28.03.22-02.04.22		
3	ІІ атестація ІІ розділ випускної роботи молодшого спеціаліста	10.05.22-14.05.22		
4	ІІІ атестація ІІІ розділ випускної роботи молодшого спеціаліста, висновки та рекомендації, додатки, реферат	30.05.22-04.06.22		
5	Перевірка випускної роботи програмою «Антиплагіат»	30.05.22-18.06.22		
6	Доопрацювання випускної роботи молодшого спеціаліста, підготовка презентації, отримання відгуку керівника і рецензії	06.06.22-11.06.22		
7	Попередній захист випускної роботи молодшого	14.06.22-18.06.22		
8	Подача випускної роботи молодшого спеціаліста на кафедру	за 3 дні до захисту		
9	Захист випускної роботи молодшого спеціаліста	20.06.22-25.06.22		

Керівник випускної роботи _____ Д.О. Костерной

(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ І.В. Агапов

(підпис студента) (прізвище та ініціали)

РЕФЕРАТ

Випускна робота містить 60 сторінок, 36 рисунків, 10 лістингів, 1 додаток.

Метою роботи є розробка прикладної програми для ветеринарної лікарні.

Об’єктом дослідження є застосування технологій для створення сучасних прикладних програм.

Предметом дослідження є створення прикладної програми для обліку пацієнтів ветеринарної клініки.

Здійснено детальний огляд предметної області та сучасних аналогів. Виявлено, що розробка прикладної програми є доцільною. Проект реалізовано за допомогою C#.

Програмний продукт є легким у використанні, має привабливий та інтуїтивно зрозумілий інтерфейс, достатній функціонал. Додаток дозволяє заощаджувати час за рахунок мінімального введення даних.

Даний програмний продукт стане в нагоді фізичним особам, підприємцям та іншим учасникам малого бізнесу.

ДОДАТОК, C#, WPF, PostgreSQL

ЗМІСТ

ПЕРЕЛІК УМОВНИК ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скорочення	Повна назва	Пояснення/переклад
ПІБ	Прізвище, ім’я, по батькові	
ПК	Персональний комп’ютер	Однокористувальницька універсальна мікроЕОМ
ОС	Операційна система	
ПЗ	Програмне забезпечення	Сукупність програм системи оброблення інформації та програмних документів
БД	База даних	Сукупність даних, що зберігаються відповідно до схеми даних, маніпуляція яких виконується відповідно до правил моделювання даних
СУБД	Система управління базами даних	
MS	Microsoft	
VS	Visual Studio	Середа програмування, розроблена компанією Microsoft
WinForms	Windows Forms	Інтерфейс програмування програм, який відповідає за графічний інтерфейс
WPF	Windows Presentation Foundation	Аналог WinForms з візуально привабливими можливостями взаємодії з користувачем, що використовує мову XAML
XAML	eXtensible Application Markup Language	Розширювана мова розмітки
XML	Extensible Markup Language	Розширювана мова розмітки
BAML	Binary Application Markup Language	Формат файлів Microsoft, якій створюється шляхом компіляції файлів XAML
CLR	Common Language Runtime	Середовище для байт-коду CIL, в який компілюються програми написані на .NET-сумісних мовах програмування
CIL	Common Intermediate Language	Проміжна мова, розроблена компанією Microsoft для платформи .NET Framework
SQL	Structured Query	Мова структурованих запитів

	Language	
MySQL		Вільна система керування реляційними базами даних, яка була розроблена компанією «ТсХ» для підвищення швидкодії обробки великих баз даних.
UML	Unified Modeling Language	Мова графічного опису для об’єктного моделювання
JSON	JavaScript Object Notation	Текстовий формат обміну даними на основі JavaScript
GIT	Global Information Tracker	Система керування версіями з розподіленою архітектурою
UI	User Interface	Користувальницький інтерфейс
WWW	World Wide Web	Розподілена система, яка надає доступ до документів розташованих на різних комп’ютерах
HTTP	Hyper Text Transfer Protocol	Протокол передачі гіпертекстових документів – протокол передачі даних, що використовується в комп’ютерних мережах
URL	Uniform Resource Locator	Уніфікований локатор ресурсів або адреса ресурсу
URI	Uniform Resource Identifier	Уніфікований ідентифікатор ресурсу, якій об’єднує абстрактний або фізичний ресурс

ВСТУП

У наш час медичні організації накопичують величезні обсяги даних. Від ефективності використання цієї інформації залежить якість медичної допомоги, рівень розвитку країни та кожного її суб’єкта.

Необхідність використання великих обсягів інформації під час вирішення діагностичних управлінських, терапевтичних, статистичних та інших завдань, обумовлює створення інформаційних систем у медичних установах.

Актуальність теми. До ветеринарних клінік звертається досить багато людей зі своїми вихованцями. Щоб знайти в архіві записи пацієнта необхідно витратити досить багато часу, саме для цього потрібна ця програма. Вона зробить роботу співробітників більш ефективною та скоротить тимчасові витрати на запис та пошук записів пацієнта в архівах.

Метою даної роботи є збір теоретичних відомостей про сучасні методи ведення обліку реєстрації пацієнтів на прийом до лікаря відповідних напрямів; огляд та аналіз технологій для реалізації програми; визначення основних фокусів при розробці; визначення оптимальних сфер використання інструментальних засобів, спираючись на аналіз процесу розробки системи обліку пацієнтів.

Методи, засоби та технології розробки.

Для створення додатку було обрано мову програмування C#, середу програмування Microsoft Visual Studio, базу даних PostgreSQL для зберігання даних.

Результатом виконання випускної роботи є реалізація додатку для ветеринарної клініки.

РОЗДІЛ 1
ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні поняття та аспекти

1.1.1 Пацієнт

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

Пацієнт – поширене поняття і потребує подальшого уточнення. У фармації більш широко використовуються терміни: споживач, відвідувач, хворий. Він як багатогранний людський фактор, зі зміною пріоритетів суспільства змінюється і пацієнт.

1.1.2 Ветеринар

Ветеринар – фахівець з повною вищою медичною освітою, який в установленому законом порядку постійно займається відновленням та підтримкою здоров’я, через профілактики, діагностики та лікування травм і захворювань. Це можливо завдяки знанням з фізіології, анатомії, науки медицини, хвороб та лікування, особистому досвіду та практиці.

До списку тварин, обов’язкових до вивчення ветеринаром входять велика та дрібна рогата худоба, коні, кішки, собаки, свині, свійський птах. Вони вивчають їх фізіологію, анатомію і біологічну хімію. Основні знання про хвороби дають вірусологія, мікробіологія, патанатомія, патофізіологія та паразитологія. Крім загальних дисциплін, лікар повинен вивчати приватні відомості про профілактику та лікування хвороб, які вони отримують з терапії, хірургії, паразитології та епізоотології.

1.1.3 Ветеринарна хірургія

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

Використовуючи методи та способи втручань, вона вирішує конкретні завдання:

- Відновлення зниженої або втраченої продуктивності тварини;
- Відновлення або поліпшення робочих якостей тварини;
- Сприяння якнайшвидшому відтворенню стада.

Хірурги виконують такі хірургічні процедури, як:

- Ендопротезування суглобів;
- Відновлення переломів;
- Стабілізація дефектів краніальних хрестоподібних зв’язків;
- Онкологічні операції;
- Лікування міжхребцевих трансплантів;
- Малоінвазивні процедури та лікування ран.

У ветеринарній хірургії застосовують неоперативні 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.

1.1.4 Ветеринарна клініка

Ветеринарна клініка – лікувально-профілактичний заклад для надання допомоги хворим тваринам на прийомі в спеціалізованому закладі [1].

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:

- Надання першої медичної допомоги;
- Проведення лабораторних аналізів;
- 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.

1.1.5 Десктопний додаток

Десктопний додаток – програма, яка встановлюється на персональному комп’ютері та працює під керуванням операційної системи.

Десктопний додаток встановлюється в систему через спеціальний інсталятор. Десктопні додатки високопродуктивні та зручні для користувачів, зазвичай такі додатки інтегровані з різними настільними та офісними програмами.

Вважається, що десктопні додатки швидше та їх функціональність ширше, але це залежить в основному від параметрів комп’ютера.

Головною особливістю десктопних додатків є можливість автономної роботи без підключення до Інтернету, але для їх оновлення без ручного перевстановлення необхідне підключення до інтернету.

Переваги десктопних додатків:

- Функціональність. Широкі можливості для функціоналу, можливо реалізувати практично будь-яку ідею. Інтерфейс буде зручним та інтуїтивно зрозумілим для користувача;
- Безпека та надійність;
- Кросплатформність. Можливість розробляти під різні операційні системи та мобільні пристрої, зробивши нативну версію;
- Швидкодійність.

Можливості десктопних додатків:

- Робота без інтернету. Додаток може працювати автономно, зберігаючи всі дані в пам’ять комп’ютера;
- Робота з інтернетом;
- Швидкий запуск. Десктопний додаток запускає системні файли с пам’яті комп’ютера не потребуючи постійні оновлення даних та завантаження параметрів із мережі;
- Інтерфейс. Дозволяє налаштувати інтерфейс під користувача;
- Використання додаткової периферії. Десктопні програми можуть мати доступ до всіх пристроїв, які підключені та підключаються до комп’ютера.

1.2 Мета програмної системи обліку пацієнтами

За допомогою медичної системи стає простіше налагодити медичні, управлінські, адміністративні та економічні потоки робочого процесу з обліку ветеринарної лікарні.

Програмні засоби полегшують роботу ветеринарам, керівникам, бухгалтерам та адміністративному персоналу. Забезпечити ветеринарну лікарню інформаційною системою як заклад медичного призначення, означає поліпшення якості лікування та обслуговування пацієнтів, мінімізування кількості паперової документації та автоматизування багато робочих місць зі сторони персоналу та ветеринару.

1.3 Основні аспекти при плануванні впровадження програмного продукту ветеринарної лікарні

Основні аспекти:

- Оцінити технічну підготовку медичного закладу: чи проведено інтернет, чи є комп’ютери у медичного та адміністративного персоналу, яка пропускна здатність інтернету, чи відповідає комп’ютер технічним нормам для роботи в медичній автоматизованій системі;
- Вибрати конкретний різновид автоматизованої системи: слід проаналізувати завдання і те які проблеми має розв’язувати розроблена система. Виходячи з цього вже можна вибрати відповідну інформаційну систему для потреб лікарні;
- Знайти постачальників та розробників автоматизованої системи, укласти з ними певний договір чи контракт, провести тренінг для медичного персоналу і навчити його користуватися системою.

На усіх етапах роботи при запускі програмного продукту система повинна працювати в штатному режимі. По роботі системи можуть бути незначні доопрацювання і адаптація.

1.4 Огляд останніх досліджень і публікацій

Система охорони здоров’я потребує сучасних інформаційних технологій. Індустріальні масштаби надання медичної допомоги, проблеми якості лікування, стандартизація медичних послуг, складні бізнес-процеси потребують впровадження комп’ютерних інформаційних технологій.

Інформаційні технології відіграють важливу роль в оптимізації діяльності та боротьбі з витратами системи охорони здоров’я. Використання інформаційних технологій стає невід'ємною частиною діяльності лікаря.

Ветеринарні клініки є одні з найбільш перспективних та швидко розвиваючих галузей, але проблема розвитку та формування клініки уповільнює зростання ринку ветеринарних послуг.

1.5 Огляд та аналіз існуючих аналогів

Розглянемо існуючі аналоги, щоб виявити переваги та недоліки вже розроблених додатків в цій сфері діяльності.

1.5.1 VetDesk

Додаток, який повністю автоматизує документообіг невеликої ветеринарної клініки.

Програма дозволяє повністю відмовитися від паперової картотеки, бланків результатів аналізів, призначень та рекомендацій клієнта. Пошук картки, аналізу або рахунку вихованця займає лічені секунди. Пацієнти залишаються задоволені моментально одержуваними документами, які можна надіслати електронною поштою або надрукувати. Автоматичні нагадування нагадають пацієнту або персоналу ветеринарної клініки про важливі події. Статистика та ведення рахунків допоможе керівнику підприємства завжди бути в курсі фінансових показників. Довідники допоможуть швидко і правильно заповнити потрібні форми.

Основні функції та можливості програми [2]:

- Картотека пацієнтів;
- Облік вакцинацій, товарів та послуг;
- Інтегровані нагадування;
- База амбулаторних прийомів;
- Запис аналізів та досліджень.

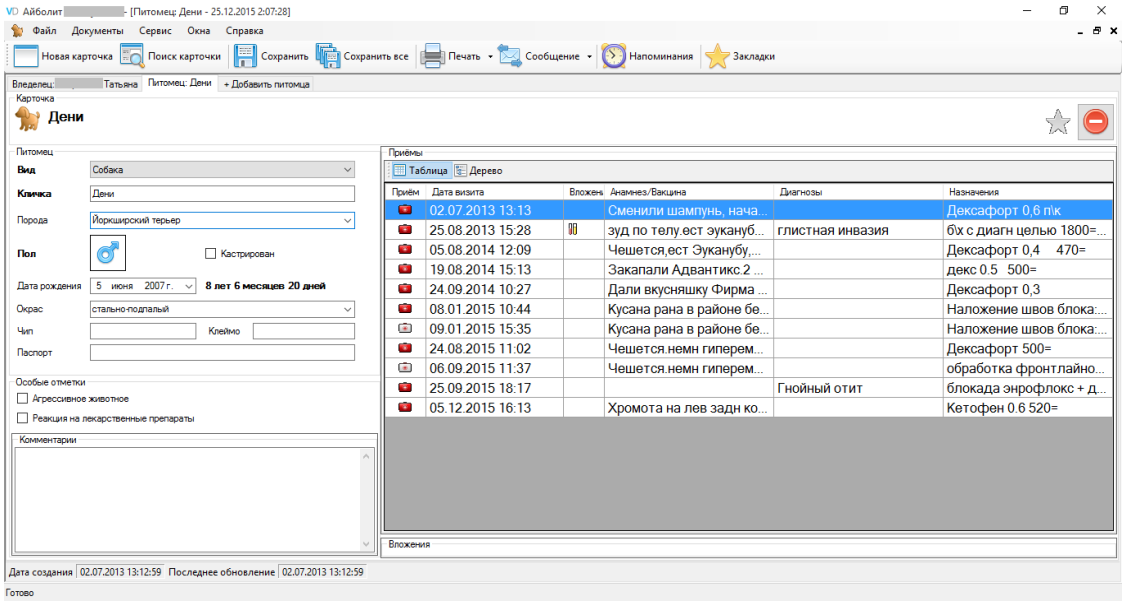

Рисунок 1.1 –Додаток «VetDesk»

1.5.2 БИТ:Айболит

Додаток для автоматизації фінансового, оперативного та складського обліку медичної діяльності та управління взаємовідносинами з клієнтами у ветеринарних клініках.

Завдяки електронним медичним карткам можна швидко переглянути результати аналізів та хвороб пацієнта. Вбудовані шаблони дозволяють ветеринару витрачати менше часу на заповнення картки клієнта.

Розділ «Ветеринарний готель» відображає оформлення та надходження вихованців до стаціонару, контроль призначення та розрахунок вартості перебування.

Розділ «Вакцинація» дозволяє сформувати нагадування клієнту про майбутні процедури-вакцинації та відображає інформацію про проведені процедури-вакцинації.

Основні функції та можливості програми [3]:

- Можливість перегляду історії хвороб та результатів аналізів у медичній картці пацієнта;
- Контроль проведених та майбутніх процедур-вакцинацій;
- Багато шаблонів для оформлення вихованців.

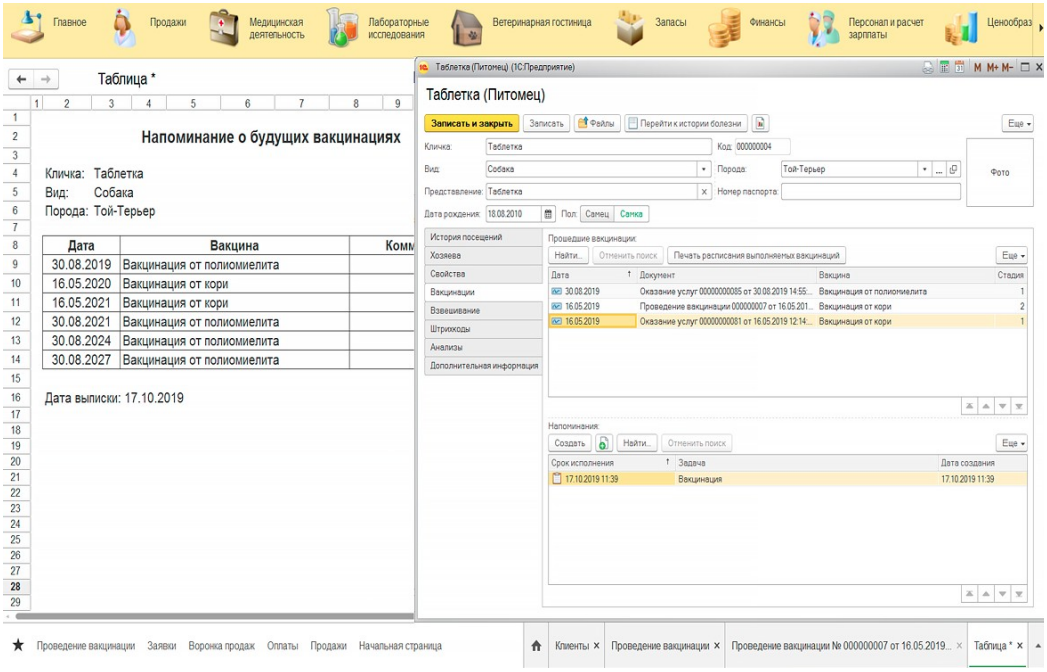

Рисунок 1.2 – Програма «БИТ:Айболит»

1.5.3 Мурмот

Додаток, який дозволяє вести облік пацієнта, список ліків та матеріалів, послуг, операцій прийому пацієнта із розрахунком вартості обслуговування та можливістю друку. Нагадування про майбутні події. Розрахунок заробітної плати.

Основні функції та можливості програми [4]:

- Довідник медикаментів, видів порід тварин, послуг у клініці та пов'язані процедури;
- Облік усіх партій медикаментів із нотифікацією про закінчення терміну придатності;
- Звіт з витрат медикаментів у прийомах, з вакцинацій;
- Розрахунок залишків за партіями медикаментів;
- Перелік аналізів із зазначенням вартості та врахуванням роздрібної націнки.

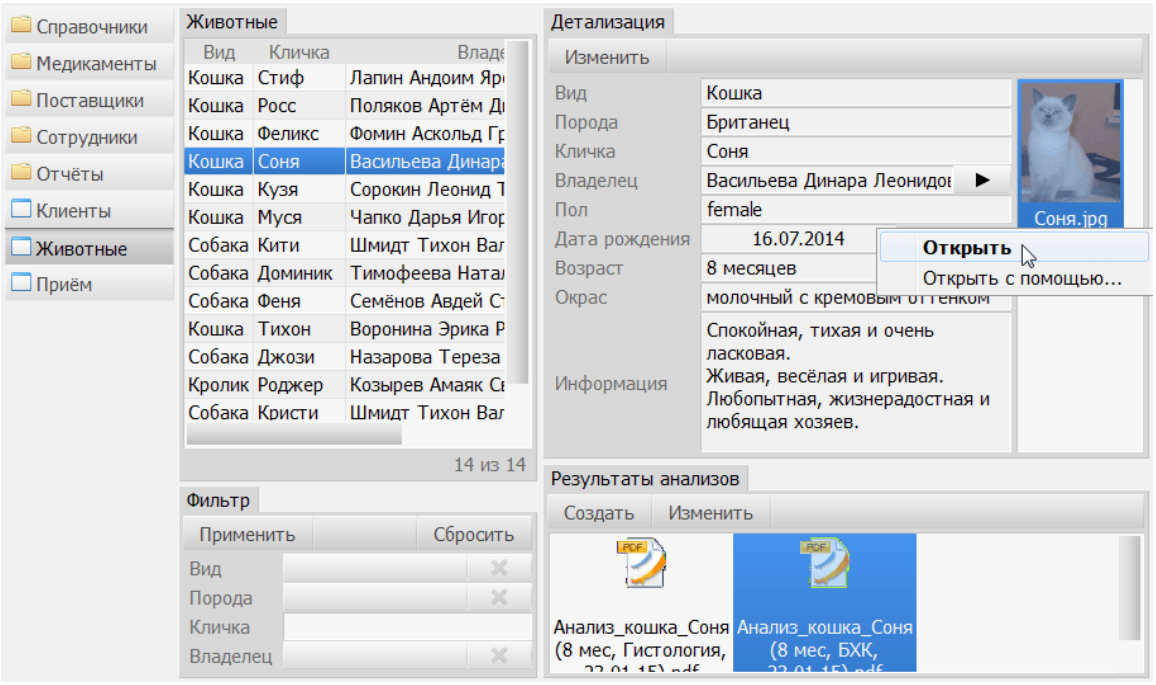

Рисунок 1.3 – Програма «Мурмот»

1.5.4 VetCliniX

Додаток обліку для ветеринарної клініки, надає можливість автоматизації, ведення обліку та зберігання результатів проведених досліджень у цифровій історії хвороби картотеці пацієнта.

Основні функції та можливості програми [5]:

- Цифрова історія хвороби;
- Картотека пацієнтів;
- Облік та зберігання результатів проведених досліджень.

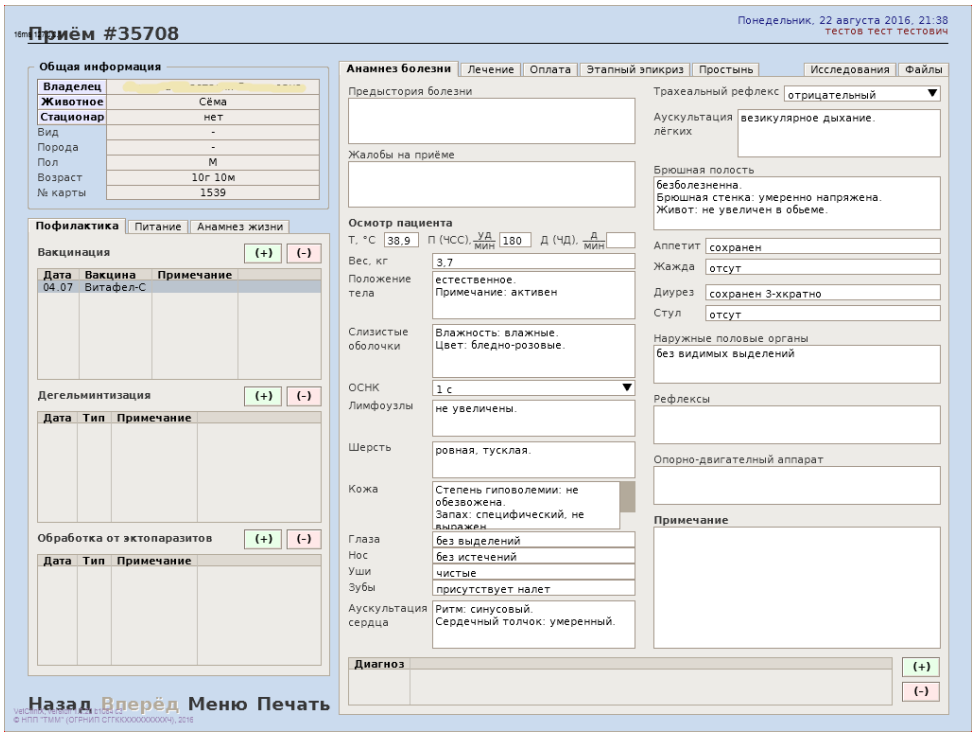

Рисунок 1.4 – Програма «VetCliniX»

1.6 Висновки за розділом

Проаналізовано вимоги та тенденції до обліку пацієнтів та лікарів з використанням помічників. Було прийнято рішення про розробку автоматизованої системи медичного закладу у вигляді прикладної програми під настільні ПК.

РОЗДІЛ 2
ІНСТРУМЕНТИ РОЗРОБКИ ДОДАТКУ

2.1 Засоби розробки

2.1.1 Мова програмування C#

На сьогоднішній день мова C# одна с найпотужніших мов, яка швидко розвівається і затребувана в IT-галузі. На ній пишуться різні програми від невеликих десктопних до веб-порталів і сервісів.

Перша версія мови C# вийшла разом із релізом MS Visual Studio .NET у лютому 2002 року. Поточна версія мови – версія C# 10.0, яка вийшла 8 листопада 2021 року разом із релізом .NET 6.

C# - об’єктно-орієнтована мова програмування із Сі-подібним синтаксисом, близький до C++ та Java. C# підтримує успадкування, поліморфізм, статичну типізацію, навантаження операторів.

Об’єктно-орієнтований підхід дозволяє вирішити з побудови великих, масштабованих, гнучких додатків, що розширюються. Ця мова насамперед розроблена для платформи .NET – середовище, що поєднує програмні технології для розробки Windows та Web додатків. Особливість мови програмування C# та фреймворку .NET – автоматичне складання сміття. Common Language Runtime викличе збирач сміття та очистить пам’ять.

Основу середовища .NET становить Common Language Runtime – середовище виконання, яка має дві основні частини:

- Ядра – набір служб, що управляються завантаженням програм, які зібрані в бібліотеці mscoree.dll;
- Бібліотеки базових класів – забрані в бібліотеці mscorlib.dll. У складі цих бібліотек виділяється Common Type System – загальна система типів та підмножина

Common Language Specification – загальномовна специфікація, що містить типи даних, які підтримуються у всіх мовах .NET.

Мова C# спеціально створена для роботи із фреймворком .NET. Фреймворк .NET – потужна платформа для створення додатків.

Основні риси платформи .NET:

- Підтримка кількох мов. Основа платформи – загальномовне середовище виконання Common Language Runtime. Завдяки їй .NET підтримує кілька мов програмування поряд з C#: C++, VB.NET, VisualBasic, J#, F#, Jscript та інші. При компіляції код інших мов компілюється у складання загальною мовою Common Intermediate Language. CIL – асемблер платформи .NET. Завдяки цьому можливо зробити окремі модулі однієї програми різними мовами програмування;
- Кросплатформність. .NET – платформа, що переноситься, але з деякими обмеженнями. .NET 6 підтримується на більшості сучасних операційних систем: Window, Linux, MacOS. При використанні різних технологій на платформі .NET є можливість розробляти програми для різних платформ на мові програмування C#: Window, Linux, MacOS, Android, iOS;
- Потужна бібліотека класів. .NET представляє єдину бібліотеку класів. Будь-яка програма на мові C# використовує бібліотеку класів .NET;
- Різноманітність технологій. Загальномовне середовище виконання (CLR) та базова бібліотека класів – основа стеку технологій. Наприклад, для роботи з базами даних призначено технологію Entity Framework Core та ADO.NET. Для побудови графічних програм з насиченим інтерфейсом використовують технологію WinUI та WPF, для простих графічних – Windows Forms. Для розробки кросплатформових десктопних та мобільних додатків використовують технологію Xamarin. Для створення веб-додатків та веб-сайтів використовують технологію ASP.NET;
- Продуктивність. Веб-програми на .NET 6 у ряді категорій випереджають веб-програми побудовані іншими технологіями.

JIT-компіляція (Just-In-Time). Код C# компілюється в збірки або додатки з розширенням dll або exe мовою CIL. При запуску на виконання додатка відбувається JIT-компіляція в машинний код, котрий потім виконується. Компілюватися буде лише частина програми до якої йде звернення. Якщо буде звернення до нескомпільованої частини, вона буде скомпільована з CIL в машинний код. Скомпільована частина програми буде зберігатися до завершення роботи програми.

Процес створення програми на мові програмування C# у середовищі .NET зображено у вигляді етапів на рисунку 2.1.

Рисунок 2.1 – Етапи створення програми на C#

Рисунок 2.2 – Логотип мови програмування C#

Відштовхувавшись від рейтингу на платформі Stack Overflow, C# займає восьме місце в рейтингу (рисунок 2.3) [6].

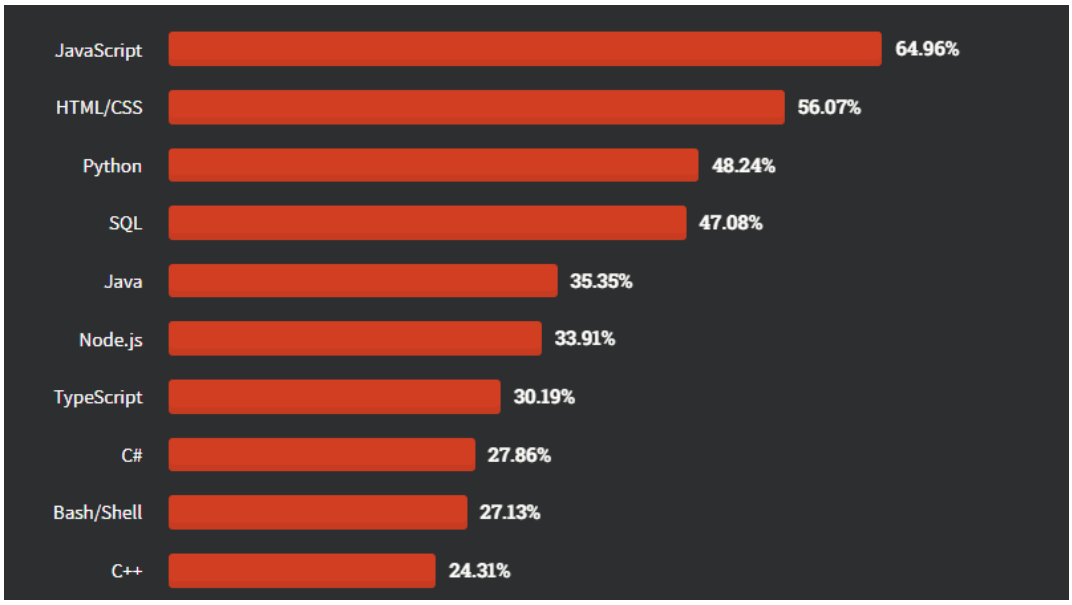

Рисунок 2.3 – Популярні мови програмування 2021 року

2.1.2 СУБД PostgreSQL

PostgreSQL – одна з найпопулярніших об’єктно-реляційних систем управління базами даних. PostgreSQL базується на мові SQL та має безліч додаткових можливостей.

Проект PostgreSQL еволюціонував із Ingres – реляційна СУБД, яка підтримується з відкритими вихідними текстами. Ingres була створена на початку 1970-х років у Каліфорнійському університеті в Берклі, закінчився на початку 1980-х років. Формально розвиток PostgreSQL почався у 1986 році під назвою POSTGRES, а в 1996 році був перейменований на PostgreSQL, що відображало акцент на SQL.

PostgreSQL надійна, безпечна та розширювальна база даних, яка має в своєму розпорядженні велику екосистему доступних засобів. Вона сумісна з усіма

основними ОС: Windows, MacOS, Linux. PostgreSQL підтримує як реляційні, так і нереляційні запити, підтримує текст, зображення, відео та звуки.

Переваги PostgreSQL:

- Доступ до потужних функцій. PostgreSQL містить багато можливостей для користувача. Можна вибрати функції попереджувальне ведення журналу, табличні простору, відновлення на час, елементи деталізованого управління доступом, вкладені транзакції, багатоваріантне управління паралелізмом та оперативне резервне копіювання;
- Відповідність вимогам та надійність. PostgreSQL відповідає властивостям узгодженості, атомарності, ізольованості та долговічності для транзакцій баз даних. Також вона підтримує кілька мов для різних атрибутів, тригерів, зовнішніх ключів, процедур та об’єднань, підтримує юнікод, багатобайтове кодування символів та міжнародні кодування;
- Ліцензія на програмне забезпечення з відкритим кодом. PostgreSQL надається з відкритим кодом, що дає більше гнучкості та можливостей для впровадження інновацій користувачам порівняно з комерційною системою баз даних;
- Масштабованість. PostgreSQL може легко управляти великим обсягом даних. Ця масштабованість стосується не тільки обсягу даних, але і кількості одночасно працюючих користувачів у ній;
- Повнотекстовий пошук та різноманітні типи індексування. PostgreSQL дає можливість різних методів індексування, включаючи індексування на основі дерев B+, узагальнене дерево пошуку та узагальнений інвертований індекс, крім повнотекстового пошуку по рядках і рядків векторних операцій;
- Гнучкість. PostgreSQL сумісна з низкою протоколів та мов програмування: C, C++, Perl, Python, Perl, Java, .NET, Ruby, Tcl та ODBC;
- Розвинена екосистема підтримки. Відкритий код PostgreSQL забезпечує підтримку спільноти розробників, які постійно удосконалюють систему;
- JSON. PostgreSQL підтримує нереляційні та реляційні запити, за допомогою виразів шляху JSON та SQL можна отримати доступ до даних JSON;

- Можливості розширення. PostgreSQL дозволяє визначати типи даних та функціональні мови, включаючи типи, що визначаються або настроюються.

Рисунок 2.4 – Логотип СУБД PostgreSQL

Відштовхувавшись від рейтингу на платформі Stack Overflow, PostgreSQL займає друге місце в рейтингу (рисунок 2.5) [6].

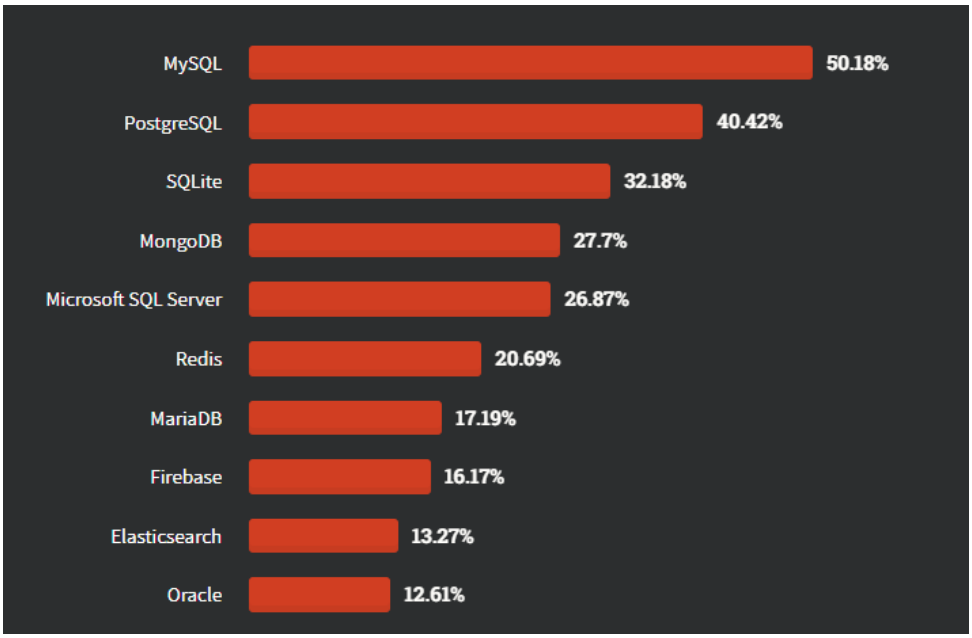

Рисунок 2.5 – Популярні СУБД 2021 року

pgAdmin – платформа з відкритим вихідним кодом для розробки та адміністрування для PostgreSQL. pgAdmin написана на мові програмування Python та jQuery, вона дозволяє виконувати конфігурування сервера PostgreSQL, обслуговування, завдання моніторингу та виконувати SQL-запити.

Особливості та можливості pgAdmin:

- Відкритий вихідний код;
- Безкоштовна;
- Можна використовувати в Windows, Linux та macOS;
- Розвантаження даних та написання SQL-запитів;
- Розробка уявлень, функцій та інших об’єктів бази даних.

2.1.3 Середа програмування Microsoft Visual Studio

MS Visual Studio – повнофункціональне інтегроване середовище розробки (IDE) з підтримкою мов програмування: C, C++, C#, F#, VB.NET, Python, JavaScript, TypeScript, XAML.

Основним розширенням файлу є SLN – Visual Studio Solution File.

Visual Studio надає сучасні інструменти для написання коду, складання, проектування графічних інтерфейсів тестування та налагодження програм, охоплюючи всі етапи розробки програмного забезпечення. Шляхом підключення потрібних розширень можна доповнити можливості Visual Studio.

Переваги Visual Studio:

- Підтримує крос-платформну розробку;
- Вбудований механізм завантаження та установки NuGet-пакетів в проект;
- Підтримує безліч мов під час розробки. Visual Studio дозволяє писати код своєю мовою або будь-якою іншою доступною, використовуючи весь час один інтерфейс (IDE);
- Інтуїтивний стиль кодування. Редактор коду підтримує вставку фрагментів коду, підсвічування синтаксису, відображення пов’язаних функцій та структури. Технологія IntelliSense допомагає істотно прискорити роботу;

- Налагоджувач. Використовується для виправлень та пошуку помилок у вихідному коді на низькому апаратному рівні. Завдяки інструментам діагностики можна оцінити якість коду з погляду використання пам’яті та продуктивності;
- Дизайнер форм. Допомагає спроектувати роботу та зовнішній вигляд кожного елементу інтерфейсу програми. Visual Studio підтримує групову розробку, дозволяючи виконувати спільне налагодження та редагування будь-якої частини коду в реальному часі. Як систему управління версіями використовує Git або Team Foundation.

Рисунок 2.6 – Логотип середи програмування Microsoft Visual Studio

Відштовхувавшись від рейтингу на платформі Stack Overflow, Microsoft Visual Studio займає друге місце в рейтингу (рисунок 2.7) [6].

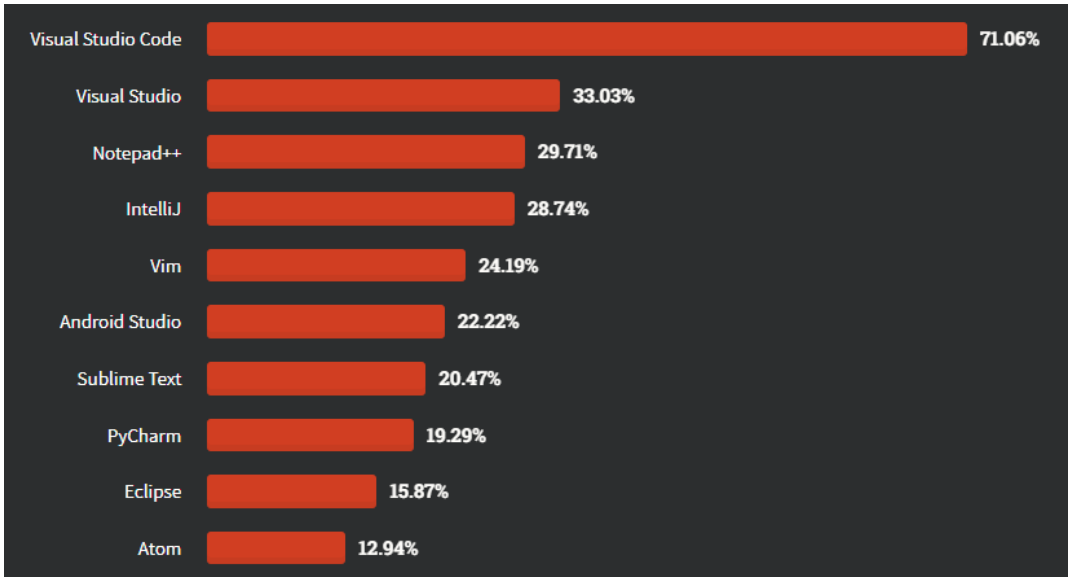

Рисунок 2.7 – Популярні середовища програмування 2021 року

2.1.4 Технологія WPF

Технологія Windows Presentation Foundation – частина екосистеми платформи .NET та підсистема для побудови графічних інтерфейсів.

Перша версія WPF 3.0 вийшла разом із .NET Framework 3.0 та ОС Windows Vista у 2006 році. Саме тоді платформа стала частиною екосистеми .NET та розвивається разом із .NET Framework. WPF підтримує Windows Vista (.NET Framework 3.5), Windows 8 (.NET Framework 4.0 та 4.5), Windows 8.1 (.NET Framework 4.5.1) та Windows 10 (.NET Framework 4.7).

При створенні програм на основі WinForms за малювання графіки та елементів керування відповідали User32 та GDI/GDI+, то WPF засновані на DirectX. Використовуючи WPF значна частина роботи з відтворення графіки лягає на процесор відеокарти, що дозволяє скористатися апаратним прискоренням графіки.

Одна з важливих особливостей WPF – використання мови декларативної розмітки інтерфейсу XAML. XAML заснований на XML, для створення насиченого графічного інтерфейсу необхідно використовувати декларативне оголошення інтерфейсу або код керованих мов C#, F#, VB.NET, або поєднувати.

eXtensible Application Markup Language – мова розмітки, яка використовується для ініціалізації об’єктів у технологія на платформі .NET.

XAML не обов’язкова частина програми, створювати елементи можливо через код мовою C#.

При компіляції програми у Visual Studio, код xaml-файлів компілюється в бінарне представлення коду Binary Application Markup Language. А код BAML у фінальне складання програми: dll-файл або exe.

Переваги XAML:

- Можливість відокремлення інтерфейсу від логіки програми. Надає можливість автономної роботи різними фахівцями: інтерфейс – дизайнери, логіка – програмісти;
- Компактність та зрозумілість коду.

Особливості WPF:

- Усталена технологія;
- Доступний конструктор;
- Підтримується у більшості версіях Windows;
- Підтримується в .NET 3.0 та пізніших версіях.

Переваги WPF:

- Використання традиційних мов програмування платформи .NET – C#, VB.NET та F# для створення логіки програми;
- Можливість декларативного визначення графічного інтерфейсу за допомогою мови розмітки XAML, яка представляє альтернативу створення елементів управління та графіки програми;
- Незалежність від роздільної здатності екрана. Всі елементи вимірюються в незалежних від пристрою одиницях. Програми легко масштабуються під різні екрани;
- Нові можливості. Прив’язка даний, створення тривимірних моделей, використання стилів, шаблонів, тем та інші;
- Взаємодія з WinForms. У додатках WPF можна використовувати елементи керування WinForms;
- Багато можливостей створення: двовимірна та тривимірна графіка, мультимедіа, анімації, прив’язка даних, шаблони, стилі, теми, можливість створення нових елементів, великий набір вбудованих елементів управління;
- Апаратне прискорення графіки. Всі компоненти програми транслюються в об’єкти, а потім візуалізуються за допомогою процесора відеокарти, що робить графіку більш плавною та підвищує продуктивність;
- Створення програм під безліч ОС сімейства Windows.

Однак WPF має певні обмеження. Для створення програм з великою кількістю тривимірних зображень краще використовувати DirectX або спеціальні фреймворки Monogame або Unity. В порівнянні з програмами на WinForms обсяг програм на

WPF та використання пам’яті в процесі роботи дещо вище. Створювати програми на WPF можна тільки під Windows.

Архітектура WPF (рисунок 2.8). WPF має два рівні: managed API та unmanaged API. Managed API містить код, який виконується під управлінням CLR.

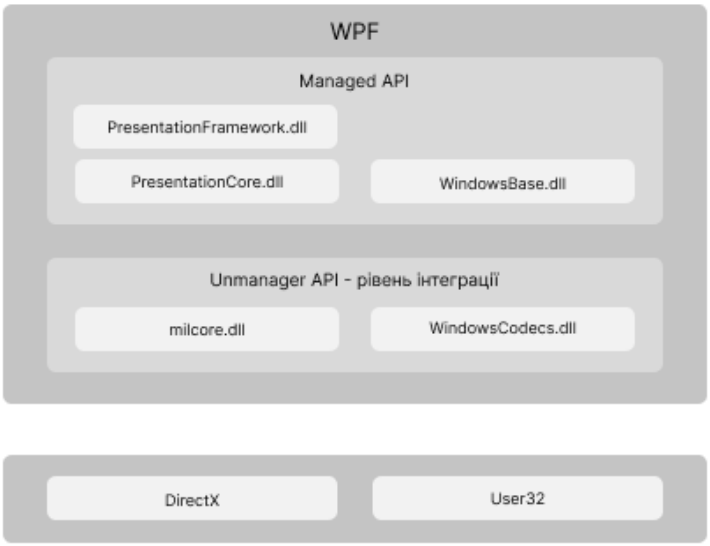

Рисунок 2.8 – Архітектура WPF

Managed API описує основний функціонал платформи WPF та складається з:

- PresentationFramework.dll – містить елементи управління та основні реалізації компонентів, які можна використовуватися при побудові графічного інтерфейсу;
- PresentationCore.dll – містить базові типи для більшості класів з PresentationFramework.dll;
- WindowsBase.dll – містить допоміжні класи, які застосовуються у WPF.

Unmanaged API використовується для інтеграції вишого рівня з DirectX:

- milcore.dll – забезпечує інтеграцію компонентів WPF із DirectX;
- WindowsCodecs.dll – надає низькорівневу підтримку для зображень у WPF.

На останньому рівні знаходяться компоненти ОС та DirectX, які відповідають за візуалізацію компонентів програми або іншу низькорівневу обробку. Завдяки низькорівневого інтерфейсу Direct3D відбувається трансляція. Для низки обчислювальних завдань, які не включають візуалізацію використовується бібліотека user32.dll.

2.2 Візуальне проектування бази даних

Необхідним етапом розробки системи є проектування бази даних. Яка забезпечить зберігання даних та наддасть можливість проводити операції над ними.

Основні етапи проектування бази даних:

- Концептуальне проектування – побудова семантичної моделі предметної області. Конкретний зміст та вигляд концептуальної моделі визначається обраним для цього формальним апаратом. Концептуальна модель включає в себе: опис понять предметної області або інформаційних об’єктів;
- Логічне проектування – створення схеми бази даних на основі, наприклад, реляційної моделі бази даних. Перетворення концептуальної моделі в логічну модель відбувається за формальними правилами;
- Фізичне проектування – створення схеми бази даних для СУБД. Специфіка СУБД може включати в себе обмеження на підтримувані типи даних та іменування об’єктів.

На етапі логічного проектування було створено зв’язок між трьома таблицями. На етапі фізичного проектування було підключено базу даних до середовища програмування MS Visual Studio та налаштовано зв’язок між базою даних та додатком.

2.3 UML-діаграма

Unified Modeling Language – уніфікована мова моделювання, використовується у парадигмі об’єктно-орієнтованого програмування, невід’ємна частина уніфікованого процесу розробки програмного забезпечення.

UML-модель – абстрактна модель система, що використовує графічні позначення. UML необхідний для візуалізації, визначення, документування та проектування програмних систем.

Діаграму прецедентів додатку для ветеринарної клініки зображено на рисунку 2.9.

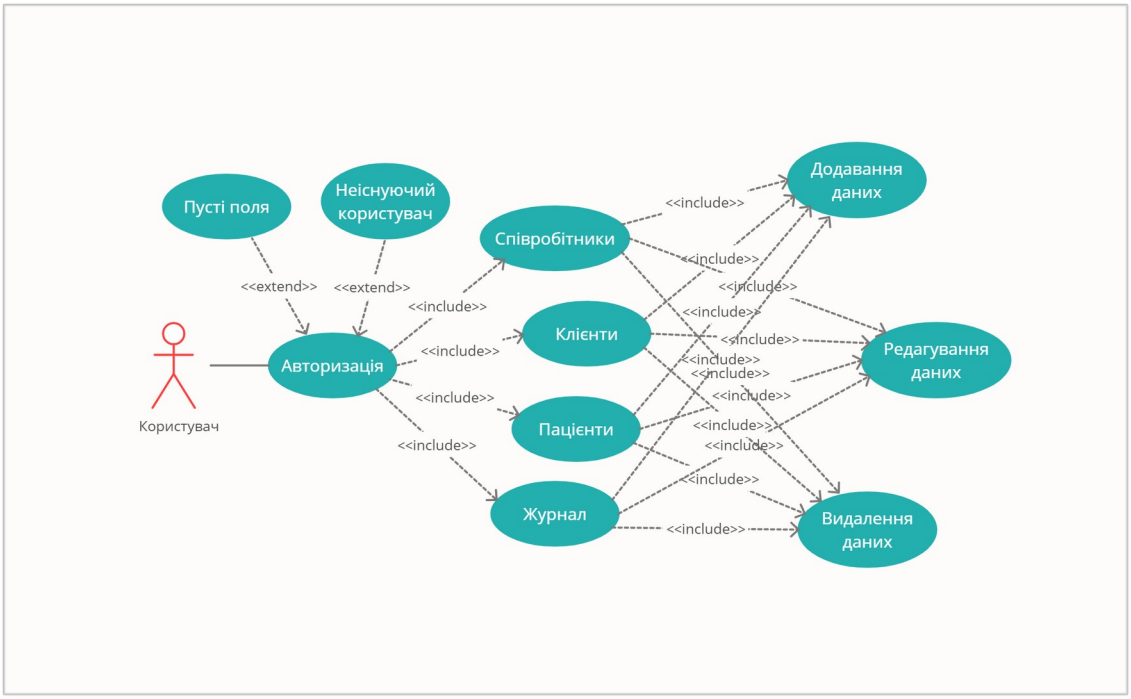

Рисунок 2.9 – Діаграма прецедентів додатку

2.4 Висновки за розділом

Проаналізувавши засоби розробки, було прийнято рішення реалізовувати проект за допомогою мови програмування C# в середовищі програмування MS Visual Studio, для збереження даних буде використано СУБД PostgreSQL.

РОЗДІЛ 3
ПРОГРАМНИЙ МОДУЛЬ

3.1 Проектування системи

На рисунку 3.1.1 зображено діаграму прецедентів. Діаграма містить одного актора - користувач.

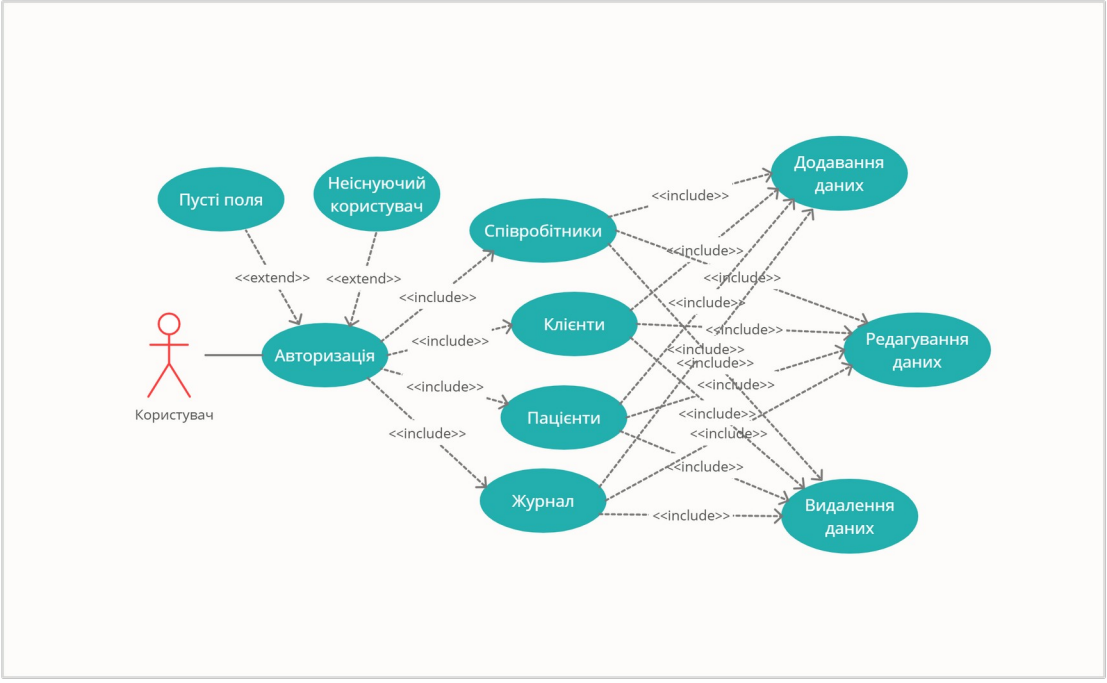

Рисунок 3.1.1 – Діаграма прецедентів

Користувач вже зареєстрований та має повний функціонал додатку, що складається з можливості редагування даних клієнтів, редагування карт пацієнтів, редагування записів відвідувань.

Для розробки додатку буде створено п’ять таблиць: авторизація, співробітники, клієнти, пацієнти та журнал. На рисунку 3.1.2 зображена ER діаграма бази даних.

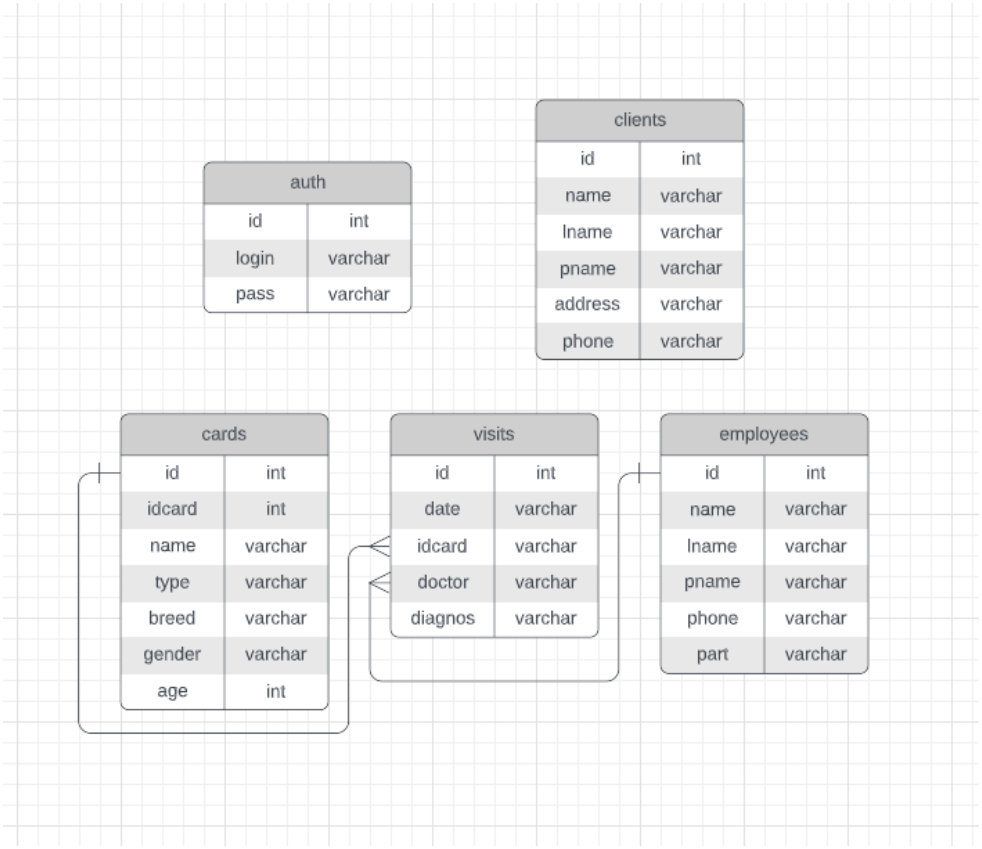

Рисунок 3.1.2 – ER діаграма бази даних

Взаємодію користувача із системою зображено на рисунку 3.1.3.

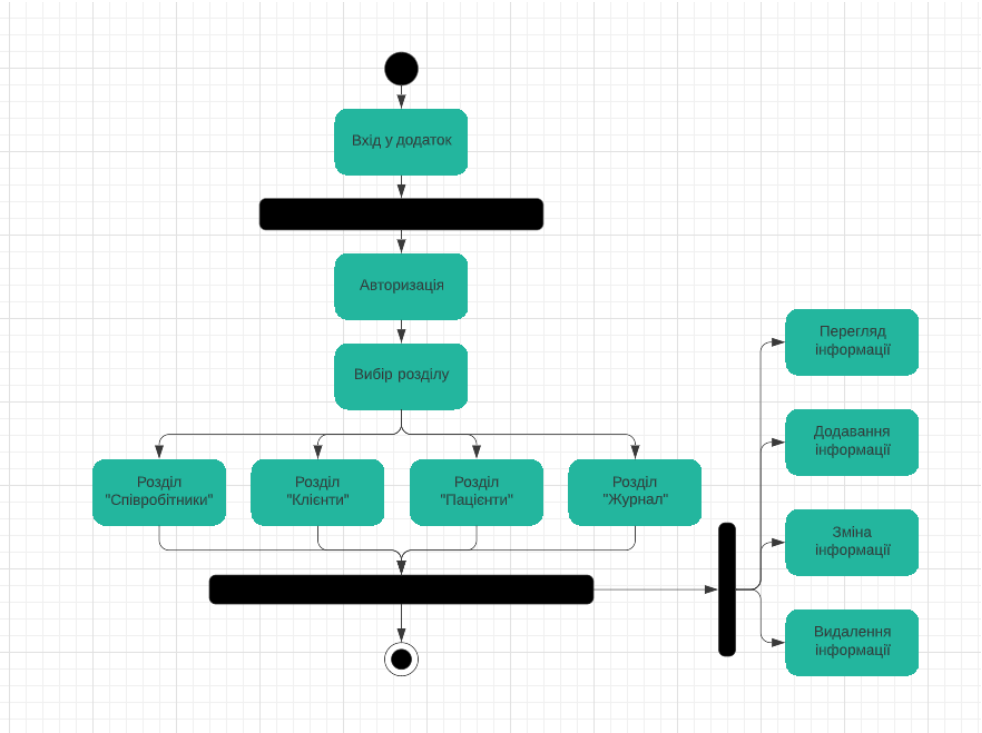

Рисунок 3.1.3 – Діаграма активності

Після входу у додаток користувачу запропоновано пройти авторизацію. Успішна авторизація направляє користувача на головне вікно додатку для вибору розділу. Користувач має можливість вибрати один с чотирьох розділів: співробітники, клієнти, пацієнти, журнал. Вибравши необхідний розділ користувач може переглянути/додати/змінити/видалити інформації розділу.

Діаграма класів проекту зображена на рисунку 3.1.4.

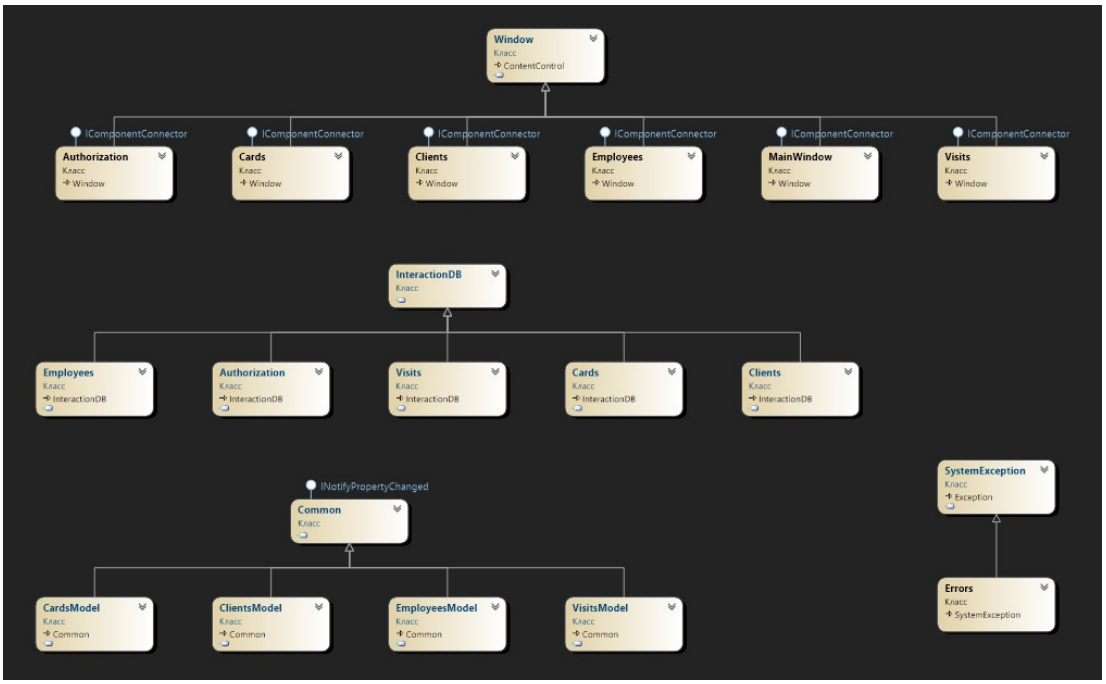

Рисунок 3.1.4 – Діаграма класів

При відкритті додатку викликається клас «Authorization». Він відповідає за авторизацію в додатку. Після успішної авторизації викликається клас «MainWindow», який відповідає за показ та вибір розділів користувача. Клас «Employees» відповідає за показ списку всіх співробітників ветеринарної лікарні. Клас «Clients» відповідає за показ списку всіх клієнтів. Клас «Cards» відповідає за показ списку всіх пацієнтів ветеринарної лікарні. Клас «Visits» відповідає за показ записів журналу. Дані класи включають клас «InteractionDB», в якому реалізовані запити до бази даних.

Клас «Errors» розширює клас помилок «SystemExeption», щоб відловлювати конкретні помилки додатку та виводити їх на екран з розшифровкою вказуючи користувачеві в чому саме помилка.

3.2 Програмування системи

3.2.1 Основні компоненти багатофайлового проекту

Під час створення додатку будо використано багатофайлову систему проекту, компоненти якої можна побачити на рисунку 3.2.1.

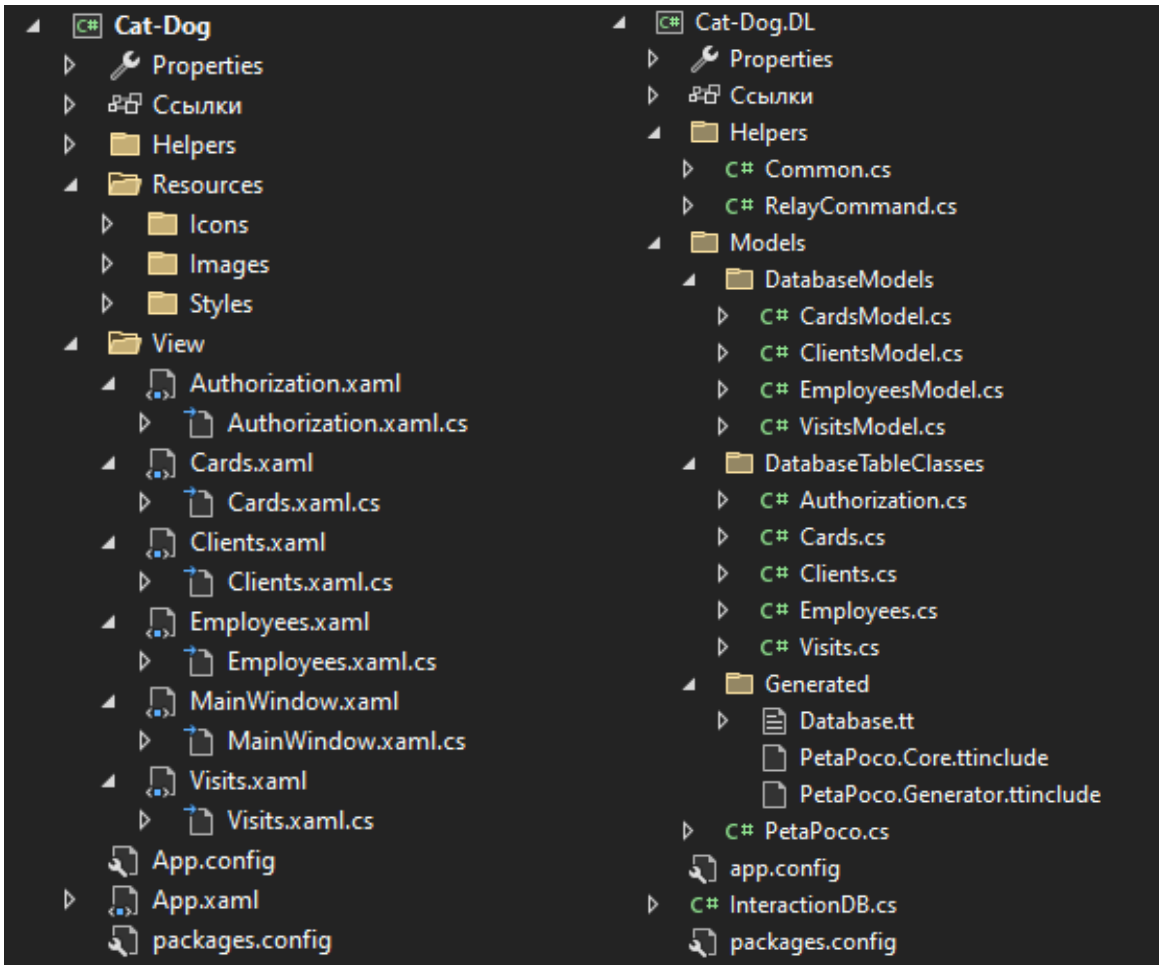

Рисунок 3.2.1 – Компоненти проекту

Розглянемо більш детальніше ці компоненти кожного підкаталогу:

- Helpers (Cat-Dog) – призначений для класів, які відстежують помилки у роботі додатку;
- Resources – зберігає в собі всі картинки, іконки та стилі, які були використані у проекті;
- View – призначений для всіх вікон проекту;
- Helpers (Cat-Dog.DL) – призначений для класів, які допомагають в реалізації моделей у каталозі «DatabaseModels»;
- DatabaseModels – призначений для класів, які використовуються інтерфейсом для відображення даних;
- DatabaseTableClasses – призначений для класів, які необхідні для отримання даних з бази даних;
- Generated -призначений для використання бібліотеки PetaPoco.

3.2.2 Розробка та створення бази даних

База даних буде складатися з п’яти таблиць: авторизації, співробітники, клієнти, пацієнти та журнал.

Таблиця авторизації повинна мати такі поля: логін та пароль. На лістингу 3.2.1 описано SQL-запит для створення таблиці авторизації. Структуру таблиці показано на рисунку 3.2.2.

Лістинг 3.2.1 - SQL-запит для створення таблиці

```
CREATE TABLE auth (
    id integer NOT NULL,
    login character varying,
    pass character varying,
    PRIMARY KEY (id)
)
```

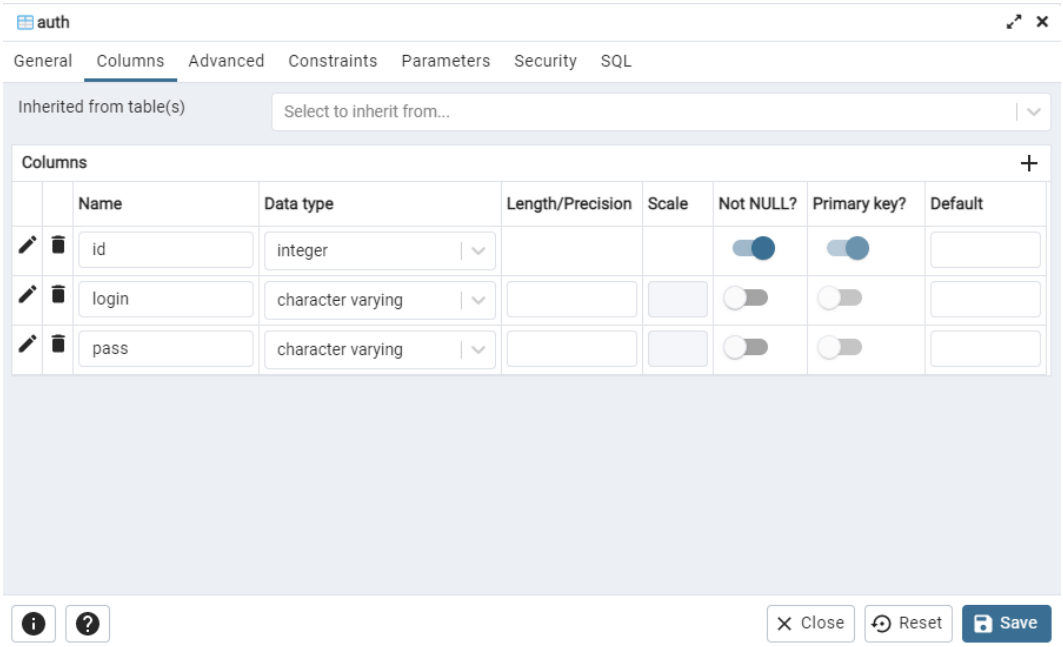

Рисунок 3.2.2 – Структура таблиці авторизації

Таблиця співробітників повинна мати такі поля: ім’я, прізвище, по батькові, номер телефону та група. На лістингу 3.2.2 описано SQL-запит для створення таблиці співробітників. Структуру таблиці показано на рисунку 3.2.3.

Лістинг 3.2.2 - SQL-запит для створення таблиці

```
CREATE TABLE employees
(
    id integer NOT NULL DEFAULT nextval('employees_id_seq'::regclass),
    name character varying,
    lname character varying,
    pname character varying,
    phone character varying,
    part character varying,
    PRIMARY KEY (id)
);
```

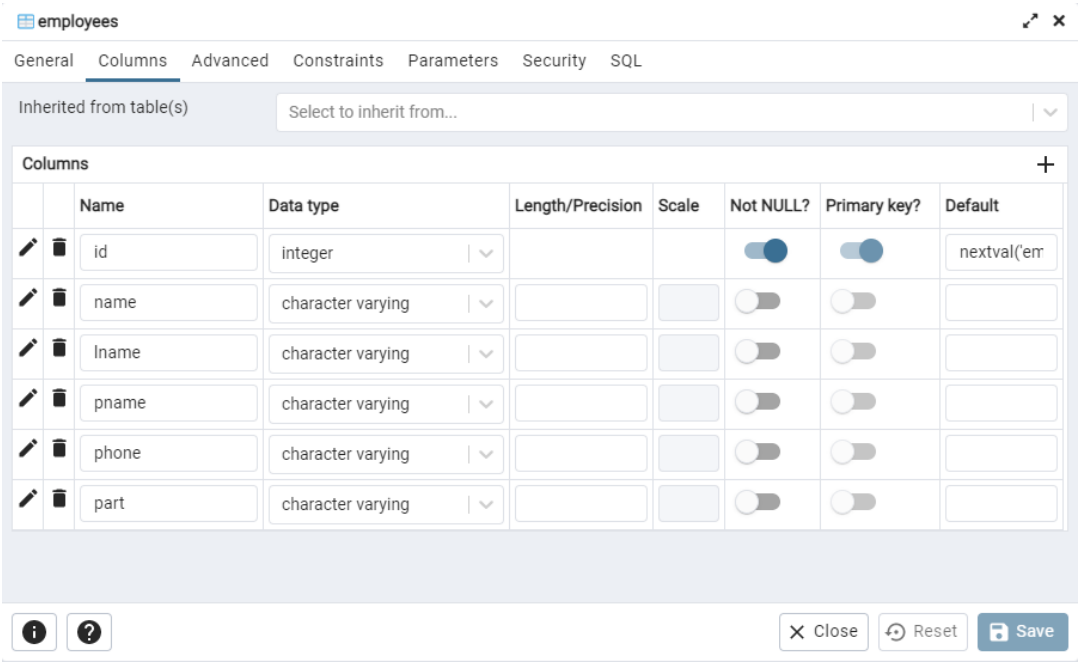

Рисунок 3.2.3 – Структура таблиці співробітників

Таблиця клієнтів повинна мати такі поля: ім’я, прізвище, по батькові, номер телефону та адреса. На лістингу 3.2.3 описано SQL-запит для створення таблиці клієнтів. Структуру таблиці показано на рисунку 3.2.4.

Лістинг 3.2.4 - SQL-запит для створення таблиці

```
CREATE TABLE clients
(
    id integer NOT NULL DEFAULT nextval('clients_id_seq'::regclass),
    name character varying,
    lname character varying,
    pname character varying,
    address character varying,
    phone character varying,
    PRIMARY KEY (id)
)
```

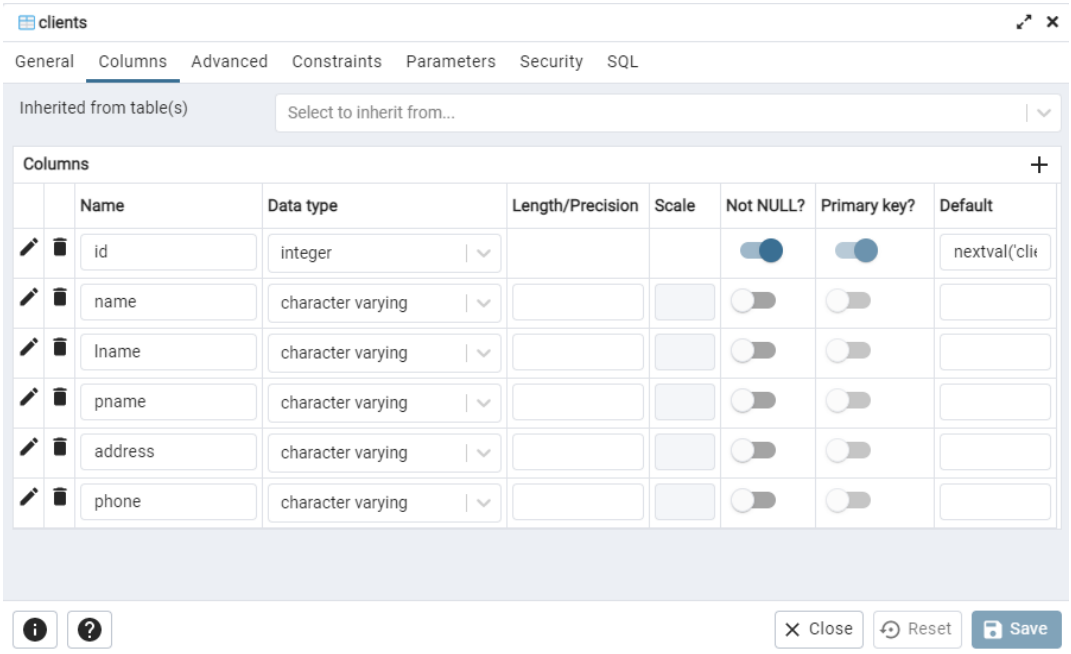

Рисунок 3.2.4 – Структура таблиці клієнтів

Таблиця пацієнтів повинна мати такі поля: номер картки, кличка, вид, порода, стать та вік. На лістингу 3.2.4 описано SQL-запит для створення таблиці пацієнтів. Структуру таблиці показано на рисунку 3.2.5.

Лістинг 3.2.4 - SQL-запит для створення таблиці

```
CREATE TABLE cards
(
    id integer NOT NULL DEFAULT nextval('cards_id_seq'::regclass),
    idcard character varying,
    name character varying,
    type character varying,
    breed character varying,
    gender character varying,
    age character varying,
    PRIMARY KEY (id)
)
```

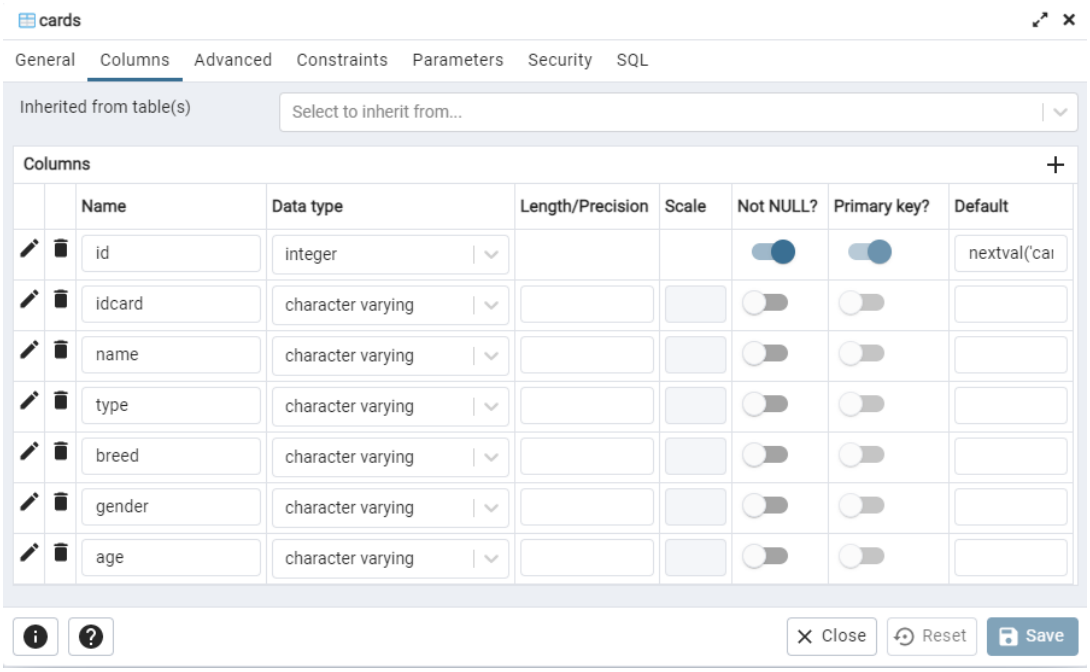

Рисунок 3.2.5 – Структура таблиці пацієнтів

Таблиця журналу повинна мати такі поля: дата, номер картки, лікар та діагноз. На лістингу 3.2.5 описано SQL-запит для створення таблиці журналу. Структуру таблиці показано на рисунку 3.2.6.

Лістинг 3.2.5 - SQL-запит для створення таблиці

```
CREATE TABLE visits
(
    id integer NOT NULL DEFAULT nextval('visits_id_seq'::regclass),
    date character varying,
    idcard character varying,
    doctor character varying,
    diagnos character varying,
    PRIMARY KEY (id)
)
```

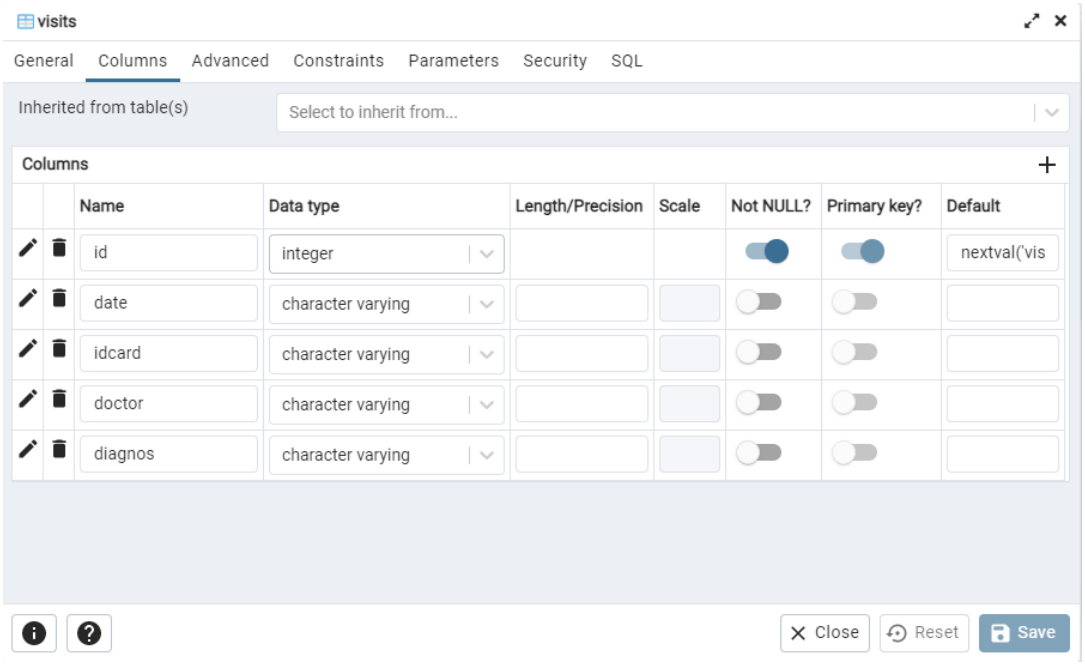

Рисунок 3.2.6 – Структура таблиці журналу

Для підключення бази даних до додатку використовується бібліотека «PetaPoco» та «Npgsql». Npgsql (Net Data Provider for PostgreSQL) необхідний для можливості працювати з базою даних PostgreSQL. PetaPoco – технологія програмування, котра пов’язує концепції об’єктно-орієнтованих мов програмування із базою даних, створюючи «віртуальну об’єктну базу даних». PetaPoco необхідний для виконання всіх операцій над таблицями бази даних.

Для підключення СУБД необхідно додати строку підключення у файл проекту «App.config» (лістинг 3.2.6).

Лістинг 3.2.6 – Код для підключення СУБД

```
<connectionStrings>
  <add name="DatabaseConnect" providerName="Npgsql"
connectionString="Server=localhost;Port=5432;Username=postgres;Password=1234;Data
base=CatDog;" />
</connectionStrings>
```

Клас «InteractionDB» є з’єднувачем між програмою та базою даних, у якому знаходяться всі зіпити до бази даних (лістинг 3.2.7).

Лістинг 3.2.7 – Код класу «InteractionDB»

```
public class InteractionDB
   {
      public bool CheckAuthorization(string login = "", string password = "")
      { return new PetaPoco.Database("DatabaseConnect").Fetch<Authorization>($"select * from auth where login = '{login}' and pass = '{password}'").Count > 0 ? true : false; }

      public List<Employees> GetListEmployees()
      { return new PetaPoco.Database("DatabaseConnect").Fetch<Employees>("select * from employees"); }

      public void AddEmployees(string name, string lName, string pName, string phone, string part)
      { try { new PetaPoco.Database("DatabaseConnect").Execute($"insert into employees(name, lname, pname, phone, part) values('{name}', '{lName}', '{pName}', '{phone}', '{part}')"); } catch (System.Exception) { } }

      public void UpdateEmployees(int id, string name, string lName, string pName, string phone, string part)
      { try { new PetaPoco.Database("DatabaseConnect").Execute($"update employees set name = '{name}', lname = '{lName}', pname = '{pName}', phone = '{phone}', part = '{part}' where id = {id}"); } catch (System.Exception) { } }

      public void DeleteEmployees(int id)
      { try { new PetaPoco.Database("DatabaseConnect").Execute($"delete from employees where id = '{id}'"); } catch (System.Exception) { } }
```

```
public List<Clients> GetListClients()
{ return new PetaPoco.Database("DatabaseConnect").Fetch<Clients>("select * from clients"); }

public void AddClients(string name, string lName, string pName, string address, string phone)
{ try { new PetaPoco.Database("DatabaseConnect").Execute($"insert into clients(name, lname, pname, address, phone) values('{name}', '{lName}', '{pName}', '{address}', '{phone}')"); } catch (System.Exception) { } }

public void UpdateClients(int id, string name, string lName, string pName, string address, string phone)
{ try { new PetaPoco.Database("DatabaseConnect").Execute($"update clients set name = '{name}', lname = '{lName}', pname = '{pName}', address = '{address}', phone = '{phone}' where id = {id}"); } catch (System.Exception) { } }

public void DeleteClients(int id)
{ try { new PetaPoco.Database("DatabaseConnect").Execute($"delete from clients where id = '{id}'"); } catch (System.Exception) { } }

public List<Visits> GetListVisits()
{ return new PetaPoco.Database("DatabaseConnect").Fetch<Visits>("select * from visits"); }

public void AddVisits(string date, string idCard, string doctor, string diagnos)
{ try { new PetaPoco.Database("DatabaseConnect").Execute($"insert into visits(date, idcard, doctor, diagnos) values('{date}', '{idCard}', '{doctor}', '{diagnos}')"); } catch (System.Exception) { } }
```

```
        public void UpdateVisits(int id, string date, string idCard, string doctor, string
diagnos)
        { try { new PetaPoco.Database("DatabaseConnect").Execute($"update visits set date
= '{date}', idcard = '{idCard}', doctor = '{doctor}', diagnos = '{diagnos}' where id =
{id}"); } catch (System.Exception) { } }

        public void DeleteVisits(int id)
        { try { new PetaPoco.Database("DatabaseConnect").Execute($"delete from visits
where id = '{id}'"); } catch (System.Exception) { } }

        public List<Cards> GetListCards()
        { return new PetaPoco.Database("DatabaseConnect").Fetch<Cards>("select * from
cards"); }

        public void AddCards(string idCard, string name, string type, string breed, string
gender, string age)
        { try { new PetaPoco.Database("DatabaseConnect").Execute($"insert into
cards(idcard, name, type, breed, gender, age) values('{idCard}', '{name}', '{type}',
'{breed}', '{gender}', '{age}')"); } catch (System.Exception) { } }

        public void UpdateCards(int id, string idCard, string name, string type, string breed,
string gender, string age)
        { try { new PetaPoco.Database("DatabaseConnect").Execute($"update cards set
idcard = '{idCard}', name = '{name}', type = '{type}', breed = '{breed}', gender =
'{gender}', age = '{age}' where id = {id}"); } catch (System.Exception) { } }

        public void DeleteCards(int id)
        { try { new PetaPoco.Database("DatabaseConnect").Execute($"delete from cards
where id = '{id}'"); } catch (System.Exception) { } }
    }
```

Для отримання кожного списку даних з бази даних необхідно створити відповідний клас. Клас для таблиці авторизації показано на лістингу 3.2.8.

Лістинг 3.2.8 – Код класу таблиці авторизації

```
public class Authorization
  {
    public int id { get; set; }
    public string login { get; set; }
    public string pass { get; set; }
  }
```

3.2.3 Розробка та створення додатку

Після запуску програми спочатку повинно відкритися вікно авторизації (рисунок 3.2.7). Для успішної авторизації користувачу необхідно ввести логін, пароль та натиснути кнопку «Увійти».

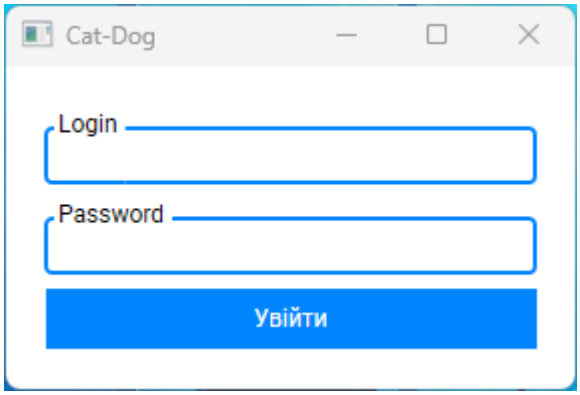

Рисунок 3.2.7 – Вікно авторизації

Якщо користувач не проходить верифікацію, буде викликано вікно помилки, приклад зображено на рисунку 3.2.8.

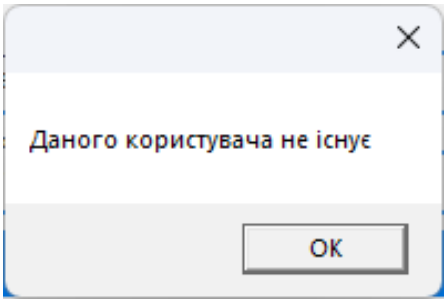

Рисунок 3.2.8 – Приклад вікна помилки

Для перехоплення помилок, які можуть виникнути під час роботи додатку необхідно створити клас помилок «Errors» (лістинг 3.2.9). У класі присутній метод, який повертає повідомлення з текстом помилки. Приклад «try-блоку» можна побачити на лістингу 3.2.10.

Лістинг 3.2.9 – Код класу «Errors»

```
class Errors : SystemException
  {
    public Errors() : base() { }

    public Errors(string message) : base(message) { }

    private void ShowErrorMessage(string error_msg)
    {
      System.Windows.MessageBox.Show(error_msg);
    }

    public void ErrorMessage()
    {
      switch (Message)
      {
        case "login_empty":
          ShowErrorMessage("Поле \"Login\" не может бути порожнім");
```

```
                break;
            case "pass_empty":
                ShowErrorMessage("Поле \"Password\" не может бути порожнім");
                break;
            case "err_auth":
                ShowErrorMessage("Даного користувача не існує");
                break;
            default:
                break;
        }
    }
}
```

Лістинг 3.2.10 – Приклад «try-блоку»

```
try
{
    if (textBoxLogin.Text == "")
        throw new Helpers.Errors("login_empty");
    else if (textBoxPassword.Text == "")
        throw new Helpers.Errors("pass_empty");
    else if (InteractionDB.CheckAuthorization(textBoxLogin.Text, textBoxPassword.Text))
    {
        MainWindow mainWindow = new MainWindow();
        mainWindow.Show();

        this.Close();
    }
    else
        throw new Helpers.Errors("err_auth");
}
```

```
catch (Helpers.Errors error)
{
    error.ErrorMessage();
}
```

У разі проходженні верифікації користувач потрапляє на головне вікно додатку (рисунок 3.2.9). Користувач має можливість відкрити один з розділів: співробітники, клієнти, пацієнти, журнал.

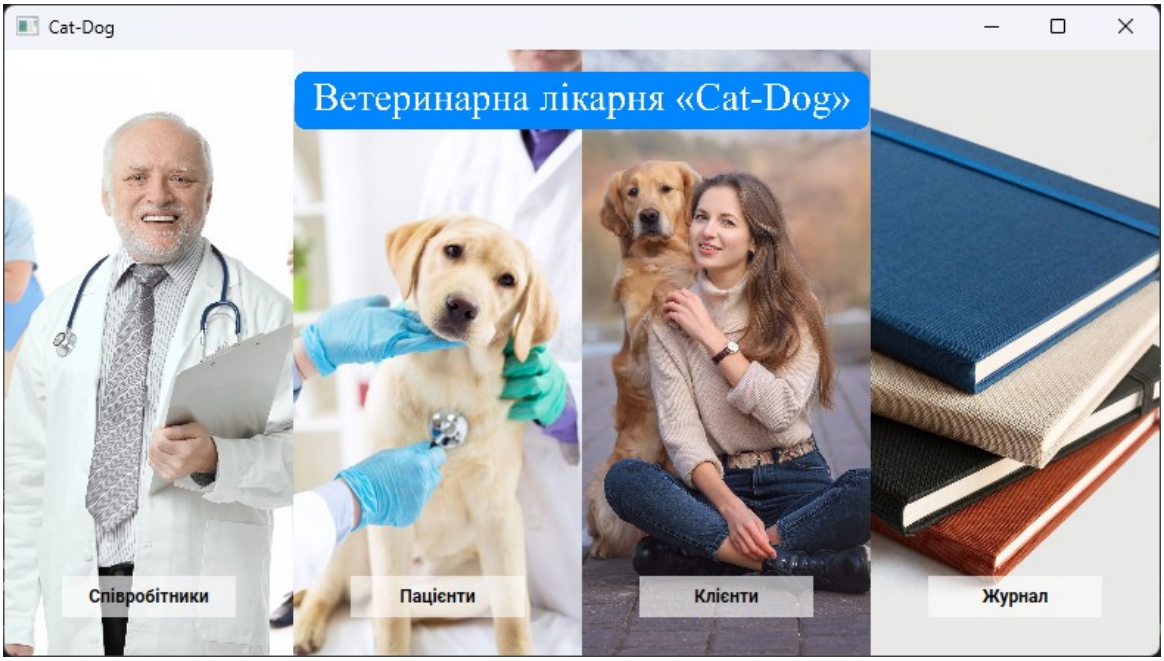

Рисунок 3.2.9 – Головне вікно додатку

Розділ «Співробітники» (рисунок 3.2.10) надає можливість користувачу побачити список всіх робітників ветеринарної лікарні. Користувач має можливість переглянути, додати, змінити або видалити данні співробітника за необхідністю.

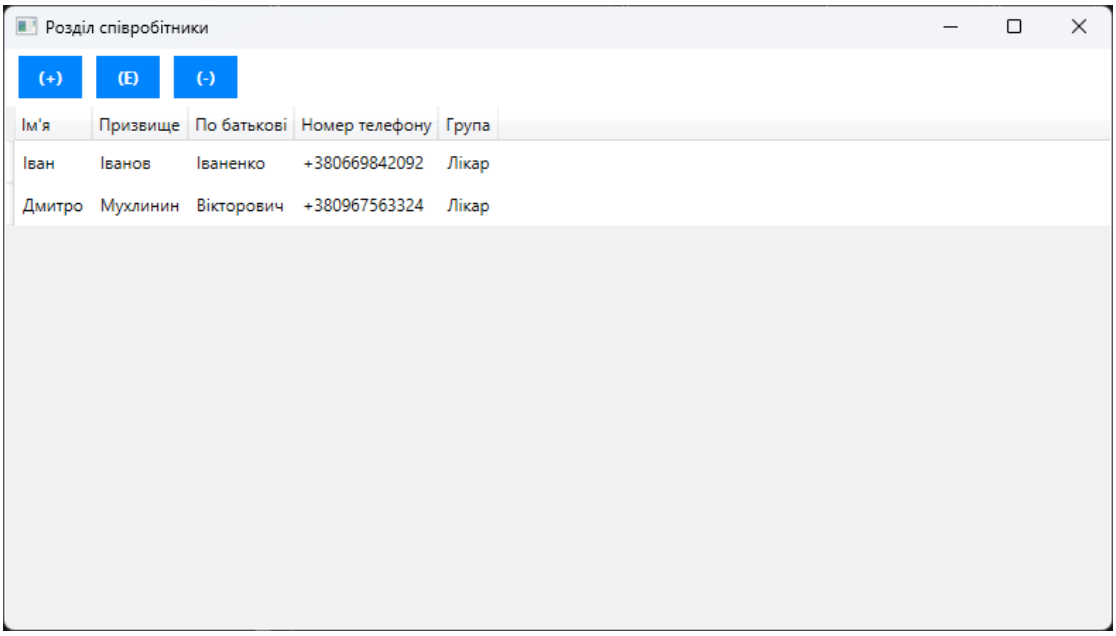

Рисунок 3.2.10 – Розділ «Співробітники»

Розділ «Клієнти» (рисунок 3.2.11) надає можливість користувачу побачити, додати, змінити або видалити данні з списку всіх клієнтів ветеринарної лікарні.

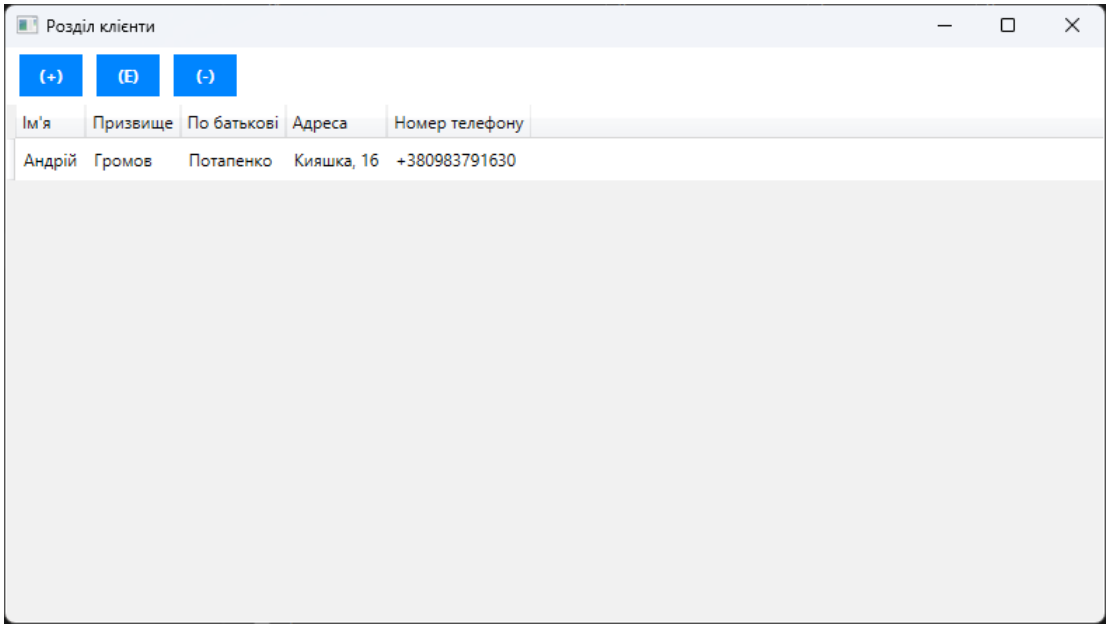

Рисунок 3.2.11 – Розділ «Клієнти»

Розділ «Пацієнти» (рисунок 3.2.12) надає можливість користувачу побачити, додати, змінити або видалити данні з списку всіх пацієнтів ветеринарної лікарні.

Для зручного пошуку користувач має можливість ввести необхідні данні у фільтри справа та для їх застосування натиснути кнопку «Застосувати».

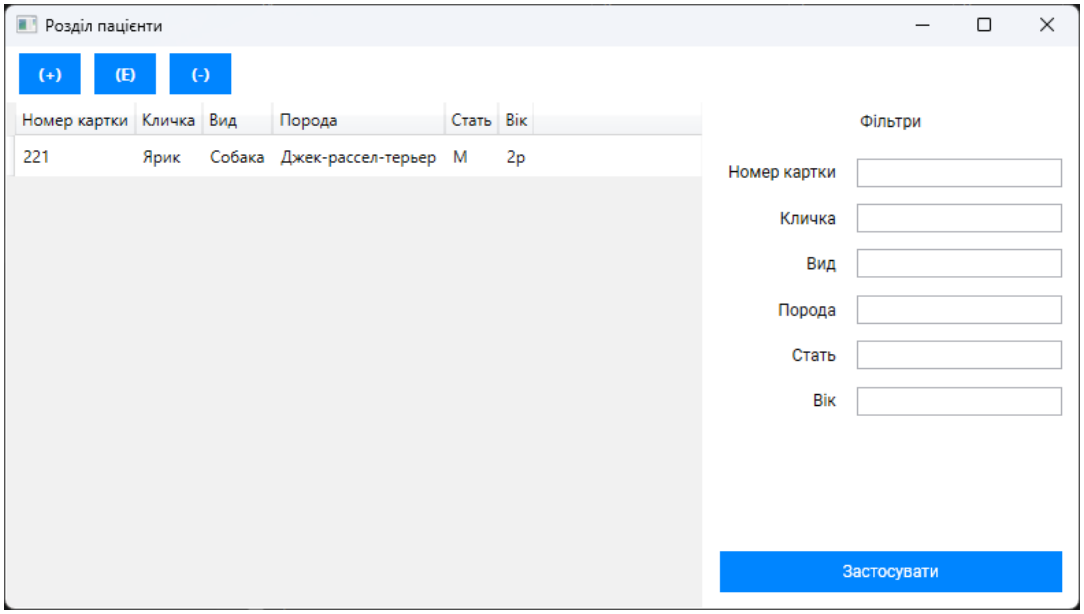

Рисунок 3.2.12 – Розділ «Пацієнти»

Розділ «Журнал» (рисунок 3.2.13) надає можливість користувачу побачити, додати, змінити або видалити данні записів журналу всіх пацієнтів ветеринарної лікарні.

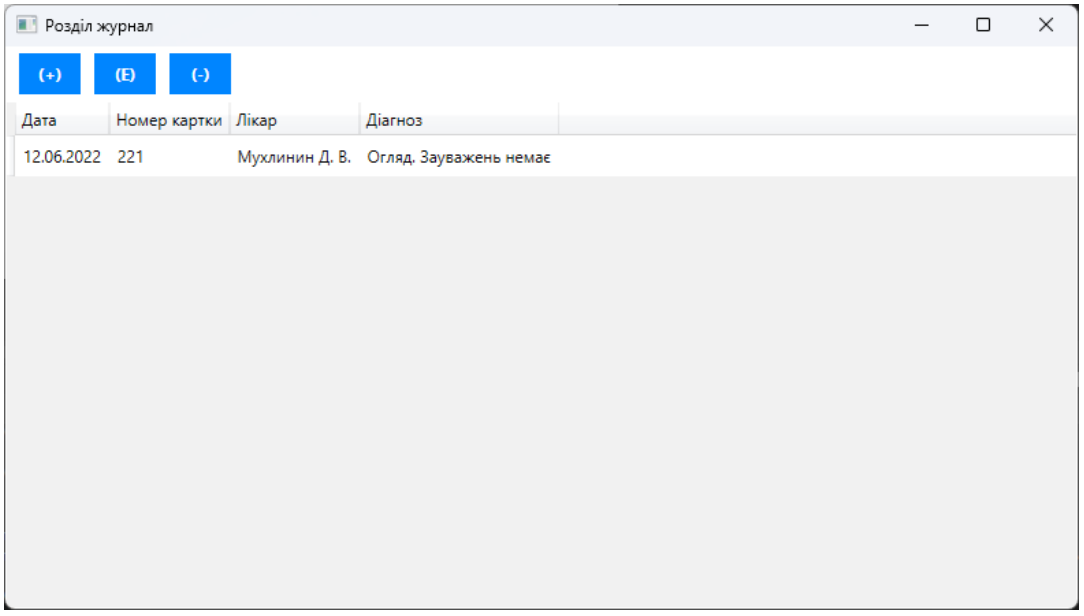

Рисунок 3.2.13 – Розділ «Журнал»

3.3 Перевірка на роботоздатність

Користувач запускаючи додаток потравляє на вікно авторизації (рисунок 3.3.1). Якщо база даних раптом не зможе працювати додаток можна запустити у тестовому режимі, якщо у поле «Login» та «Password» ввести «admin». Під час тестування авторизації помилок не було виявлено.

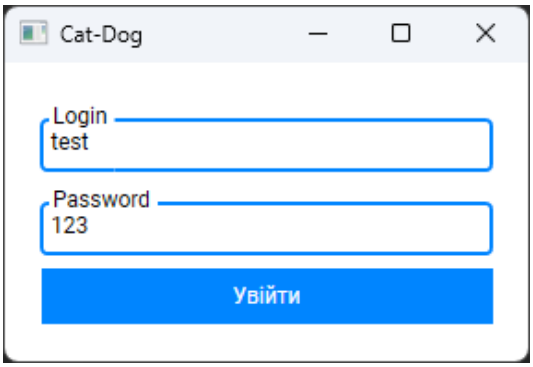

Рисунок 3.3.1 – Результат запуску додатку

Після верифікації свого аккаунту він повинен потрапити на головне вікно вибору розділів (рисунок 3.3.2).

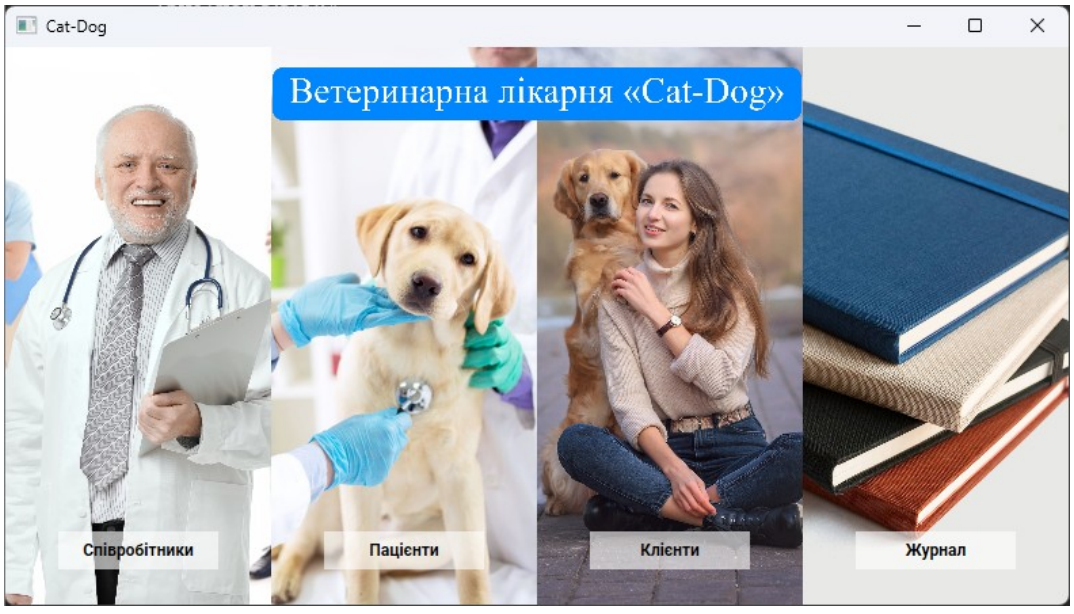

Рисунок 3.3.2 – Результат верифікації аккаунту

Перевіримо розділ «Співробітники» на працездатність перегляду, редагування, зміни, видалення інформації (рисунок 3.3.3). Спробуємо видалити одну запис з списку. Під час тестування розділу «Співробітники» помилок не було виявлено.

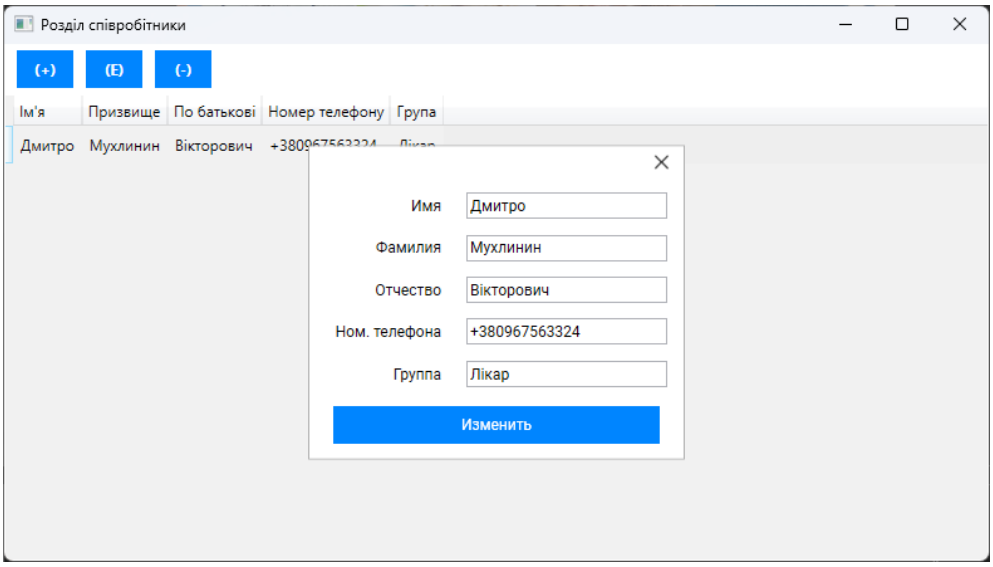

Рисунок 3.3.3 – Розділ «Співробітники»

Перевіримо розділ «Пацієнти» на працездатність перегляду, редагування, зміни, видалення та пошук інформації за допомогою фільтрів (рисунок 3.3.4). Спробуємо додати у базу даних нового пацієнта. Під час тестування розділу «Пацієнти» помилок не було виявлено.

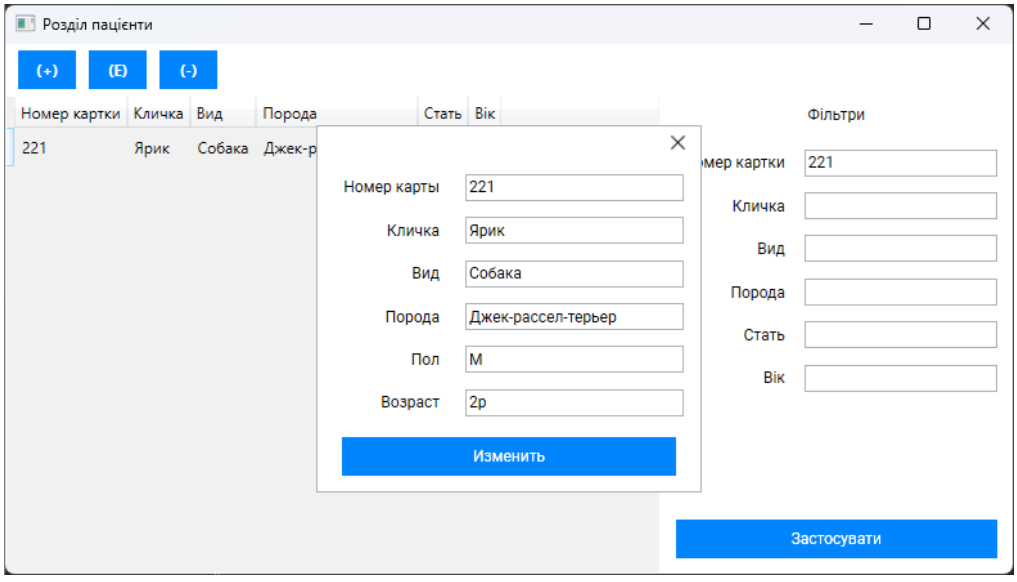

Рисунок 3.3.4 – Розділ «Пацієнти»

Перевіримо розділ «Клієнти» на працездатність перегляду, редагування, зміни та видалення інформації (рисунок 3.3.5). Спробуємо змінити адресу клієнта на нову. Під час тестування розділу «Клієнти» помилок не було виявлено.

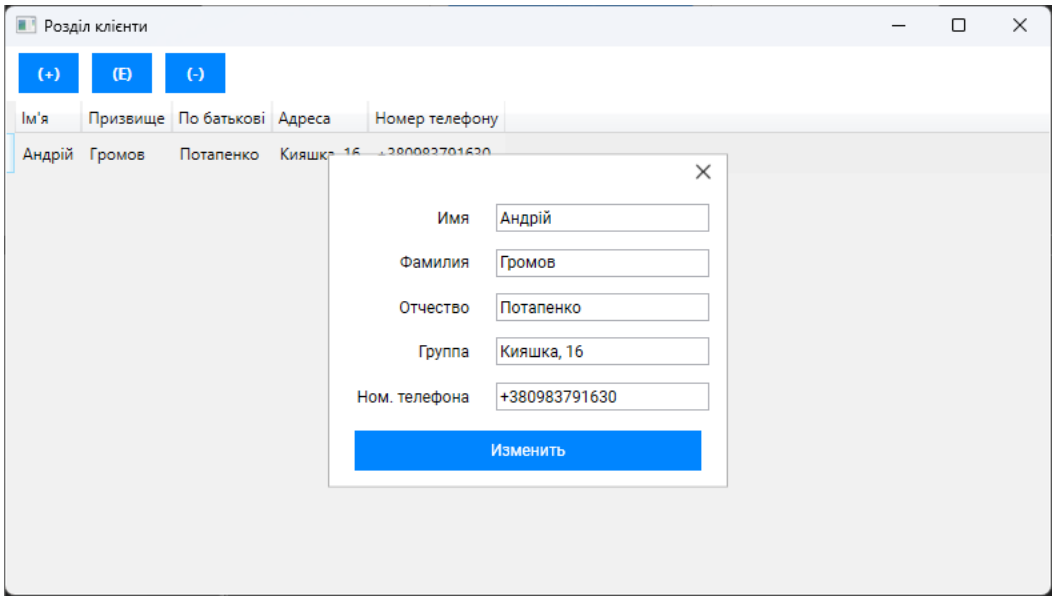

Рисунок 3.3.5 – Розділ «Клієнти»

Перевіримо розділ «Журнал» на працездатність перегляду, редагування, зміни, видалення інформації (рисунок 3.3.6). Спробуємо додати новий запис відвідування пацієнта у журнал. Під час тестування розділу «Журнал» помилок не було виявлено.

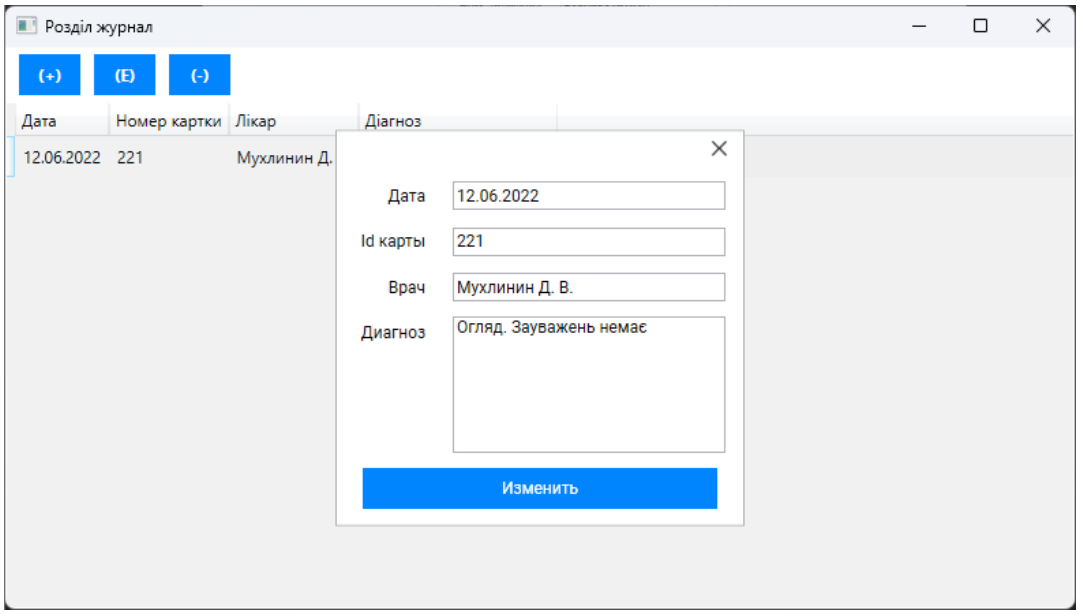

Рисунок 3.3.6 – Розділ «Журнал»

Під час тестування додатку критичних помилок не було виявлено. Вся інформація відображається коректно.

Було зауважено деякі незручності в введені даних та немає вікна підтвердження для додавання, зміни та видалення інформації. Всі ці незручності необхідно виправити у новій версії цього додатку.

3.5 Запланований функціонал

На початку проектування був запланований, але не реалізований весь додатковий функціонал додатку таки як:

- Можливість адміністрування;
- Можливість реєстрування аккаунту;
- Розділ «Склад»;
- Можливість враховувати вартість послуг;
- Можливість розрахунку заробітної плати;
- Календарний графік кожного лікаря;
- Розділ «Журнал записів»;
- Розділ «Дослідження».

3.4 Висновки за розділом

Було запроектовано, створено та підключено СУБД PostgreSQL.

Було запроектовано, проаналізовано та реалізовано функціонал додатку для ветеринарної клініки.

При тестуванні додатку отримано адекватну і стабільну відповідь програми, критичних помилок не було виявлено. Були зауважені недоліки додатку, які необхідно виправити у майбутніх версіях.

ВИСНОВКИ

В ході виконання випускної роботи молодшого спеціаліста було досягнуто поставлених задач та цілей.

Було проаналізовано вимоги та тенденції до обліку пацієнтів та лікарів з використанням помічників. Було прийнято рішення про розробку автоматизованої системи медичного закладу у вигляді прикладної програми під настільні ПК.

Проаналізовано засоби розробки та прийнято рішення реалізовувати проект за допомогою мови програмування C# в середовищі програмування MS Visual Studio, для збереження даних буде використано СУБД PostgreSQL.

Було запроектовано, створено та підключено СУБД PostgreSQL. Було запроектовано, проаналізовано та реалізовано функціонал додатку для ветеринарної клініки.

При тестуванні додатку отримано адекватну і стабільну відповідь програми, критичних помилок не було виявлено. Були зауважені недоліки додатку, які необхідно виправити у майбутніх версіях.

При написання випускної роботи молодшого спеціаліста були виконані наступні завдання:

- Досліджено літературу стосовно створення додатку;
- Виконано огляд мови програмування, середовища розробки, бази даних;
- Запроектовано та створено додаток для ветеринарної лікарні;
- Протестовано додаток на виявлення помилок та незручностей.

ПЕРЕЛІК ПОСИЛАНЬ

1. Ветеринарна клініка. [Електронний ресурс]. – Режим доступу: www. URL:https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%82%D0%B5%D1%80%D0%B8%D0%BD%D0%B0%D1%80%D0%B8%D1%8F

2. VetDesk. [Електронний ресурс]. – Режим доступу: www. URL: http://vetdesk.ru/

3. БИТ:Айболит. [Електронний ресурс]. – Режим доступу: www. URL: https://www.bit-medic.ru/bit-aybolit/

4. Мурмот. [Електронний ресурс]. – Режим доступу: www. URL: http://murmot.ru/

5. VetCliniX. [Електронний ресурс]. – Режим доступу: www. URL: https://vetclinix.ru/

6. Stack Overflow Developer Survey 2021. [Електронний ресурс]. – Режим доступу: www. URL: https://insights.stackoverflow.com/survey/2021

7. UML. [Електронний ресурс]. – Режим доступу: www. URL: https://uk.wikipedia.org/wiki/Unified_Modeling_Language

8. Інформаційний портал Microsoft Docs C#. [Електронний ресурс]. – Режим доступу: www. URL: https://docs.microsoft.com/en-us/dotnet/csharp/

9. Інформаційний портал Microsoft Docs WPF#. [Електронний ресурс]. – Режим доступу: www. URL: https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/?view=netdesktop-6.0

10. Інформаційний портал Metanit C#. [Електронний ресурс]. – Режим доступу: www. URL: https://metanit.com/sharp/

11. Інформаційний портал Harb C#. [Електронний ресурс]. – Режим доступу: www. URL: https://habr.com/ru/hub/csharp/

12. Посібник з WPF. [Електронний ресурс]. – Режим доступу: www. URL: https://metanit.com/sharp/wpf/

13. Паттерн MVVM. [Електронний ресурс]. – Режим доступу: www. URL: https://metanit.com/sharp/wpf/22.1.php

14. Паттерн MVVM. [Електронний ресурс]. – Режим доступу: www. URL: https://habr.com/ru/post/338518/

15. PetaPoco. [Електронний ресурс]. – Режим доступу: www. URL: https://github.com/CollaboratingPlatypus/PetaPoco/wiki

16. PetaPoco. [Електронний ресурс]. – Режим доступу: www. URL: https://russianblogs.com/article/35231377149/

17. PostgreSQL. [Електронний ресурс]. – Режим доступу: www. URL: https://ru.wikipedia.org/wiki/PostgreSQL

18. PostgreSQL. [Електронний ресурс]. – Режим доступу: www. URL: https://www.postgresql.org/

19. PostgreSQL. [Електронний ресурс]. – Режим доступу: www. URL: https://postgrespro.ru/docs/postgresql/9.6/index

20. pgAdmin 4. [Електронний ресурс]. – Режим доступу: www. URL: https://www.pgadmin.org/docs/pgadmin4/development/index.html

21. XAML. [Електронний ресурс]. – Режим доступу: www. URL: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-6.0

ДОДАТОК А
ЛІСТИНГ КОДУ