Kolegium Nauk
Przyrodniczych Uniwersytet
Rzeszowski

Przedmiot:

Programowanie obiektowe 2

Dokumentacja aplikacji do zarządzania kliniką weterynaryjną

Wykonała:

Natalia Skorupa 131511

Prowadzący: Mgr. Wojciech Gałka

Rzeszów 2025

Spis treści

Wstęp

Aplikacja **VetClinic** to system wspomagający zarządzanie kliniką weterynaryjną, stworzony jako aplikacja desktopowa z wykorzystaniem platformy .NET i Windows Forms. Jej głównym celem jest umożliwienie sprawnego rejestrowania i przeglądania wizyt zwierząt, obsługi danych pacjentów (zwierząt), lekarzy, leków oraz prowadzenia historii leczenia. System działa w oparciu o relacyjną bazę danych PostgreSQL, do której dostęp realizowany jest z wykorzystaniem Entity Framework Core.

Struktura aplikacji obejmuje formularze użytkownika, które umożliwiają dodawanie i edycję wizyt (`VisitAddForm`), dynamiczne wyszukiwanie zwierząt, przypisywanie lekarzy na podstawie specjalizacji, a także wybór leków stosowanych w terapii. Klasa `Constants` gromadzi wspólne dane aplikacji, takie jak lista typów zwierząt, dostępne diagnozy, lista leków, a także metoda pomocnicza do walidacji e-maila i konfiguracja połączenia z bazą danych.

Aplikacja została zaprojektowana z uwzględnieniem mechanizmów walidacji danych wprowadzanych przez użytkownika oraz ograniczeń systemowych, takich jak limit wizyt dziennych przypadających na jednego lekarza czy zakaz rezerwacji wizyt w weekendy. Komponenty interfejsu są dynamicznie konfigurowane na podstawie danych wprowadzonych lub wybranych przez użytkownika, co pozwala na bardziej elastyczną i intuicyjną obsługę systemu.

Punkt wejścia aplikacji znajduje się w klasie `Program`, gdzie inicjalizowane są dane w bazie poprzez `DbSeeder`, a następnie uruchamiany jest główny formularz interfejsu użytkownika `MainForm`. Architektura projektu pozwala na rozbudowę funkcjonalności oraz dalszą integrację z dodatkowymi modułami systemu weterynaryjnego.

Baza Danych

DBSettings

Reprezentuje ustawienia połączenia z bazą danych. Zawiera informacje potrzebne do nawiązania połączenia, takie jak host, port, nazwa bazy, użytkownik i hasło.

Lek

Model leku dostępnego w klinice. Przechowuje nazwę oraz ilość leku i posiada relację z wizytami, w których lek został wykorzystany. Używana do zarządzania zapasami i przypisaniami leków do wizyt.

Lekarz

Reprezentuje lekarza weterynarii. Dziedziczy dane osobowe z klasy `Osoba` i dodaje specjalizację oraz tryb pracy. Używana do przypisywania wizyt oraz prezentacji danych kadry lekarskiej.

Osoba

Model bazowy dla osób w systemie. Zawiera dane osobowe takie jak imię, nazwisko, email, telefon oraz datę urodzenia. Wykorzystywana jako baza dla innych modeli dziedziczących (np. lekarz, właściciel zwierzęcia).

Wizyta

Reprezentuje wizytę w klinice weterynaryjnej. Powiązana z lekarzem, zwierzęciem oraz przypisanymi lekami. Przechowuje datę wizyty i jej opis. Kluczowa dla rejestracji historii leczenia.

Zamowienie

Reprezentuje zamówienie na lek. Zawiera informacje o zamawianym leku, ilości i dacie zamówienia. Służy do zarządzania dostawami i uzupełnianiem zapasów.

Zwierze

Model opisujący zwierzę będące pod opieką kliniki. Zawiera dane takie jak imię, gatunek, typ, wiek oraz odniesienie do właściciela (`Osoba`). Używany do identyfikacji pacjentów i powiązań z wizytami.

Seedery

DbSeeder

Główna klasa odpowiedzialna za inicjalizację bazy danych.
Funkcja `SeedDatabase` sprawdza, czy są oczekujące migracje, a następnie, jeśli to konieczne, stosuje je i uzupełnia bazę testowymi danymi. Działa krok po kroku, dodając osoby, lekarzy, zwierzęta, leki, wizyty i zamówienia, jeśli baza jest pusta.

LekarzSeeder

Udostępnia przykładowych lekarzy w postaci statycznej listy. Dane zawierają podstawowe informacje (imię, nazwisko, email, specjalizacja, tryb pracy). Używana w `DbSeeder`, jeśli tabela lekarzy jest pusta.

LekSeeder

Generuje przykładowe leki z użyciem biblioteki Faker. Tworzy listę obiektów klasy `Lek` na podstawie predefiniowanej listy nazw leków (`Constants.Medicines`) i przypisuje losową ilość dla każdego.

OsobaSeeder

Tworzy zestaw testowych danych osobowych. Generuje losowe osoby z użyciem Faker, pomijając identyfikatory przypisane lekarzom. Dodaje też specjalną osobę reprezentującą centrum opieki. Przeznaczona do wypełnienia tabeli `Osoby`.

WizytaSeeder

Odpowiada za generowanie przeszłych i przyszłych wizyt na podstawie istniejących danych w bazie (zwierzęta, lekarze, leki). Dobiera lekarzy zgodnie z typem zwierzęcia i dba o ograniczenie liczby wizyt dziennych na lekarza (maks. 5). Tworzy zróżnicowany zbiór wizyt z opisem i listą leków.

ZamowienieSeeder

Tworzy losową listę zamówień na leki. Generuje ilość, datę i przypisanie leku przy użyciu Faker. Liczba rekordów jest parametryzowana.

ZwierzeSeeder

Generuje losowe zwierzęta z odpowiednim dopasowaniem gatunku do typu zwierzęcia. Każde zwierzę jest powiązane z właścicielem (z wyjątkiem zwierząt bez właściciela – np. z typem przypisanym do centrum opieki). Używa Faker do wygenerowania danych takich jak imię, wiek i właściciel.

Context

AppDbContext

Klasa reprezentująca kontekst bazy danych dla aplikacji VetClinic. Odpowiada za mapowanie modeli domenowych na tabele w bazie danych oraz konfigurację relacji między nimi. Zawiera zestawy danych (`DbSet`) dla encji: wizyty, osoby, lekarze, zwierzęta, zamówienia oraz leki.

W metodzie `OnModelCreating` definiuje:

- relację wiele-do-wielu między wizytami a lekami za pomocą tabeli pośredniej,
- typ kolumny daty wizyty jako `date`,
- relację jeden-do-wielu między wizytą a zwierzęciem z ustawieniem zachowania usunięcia na `SetNull`, co oznacza, że usunięcie zwierzęcia nie spowoduje usunięcia powiązanych wizyt, a jedynie wyczyszczenie klucza obcego.

AppDbContextFactory

Fabryka tworząca instancje `AppDbContext` na potrzeby narzędzi projektowych Entity Framework Core (np. migracji). Umożliwia utworzenie kontekstu na podstawie konfiguracji połączenia pobranej z pliku `appsettings.json` lub z przekazanego connection stringa.

Zapewnia elastyczność i ułatwia integrację z narzędziami EF Core, pozwalając na dynamiczne tworzenie kontekstu w różnych środowiskach.

Widoki

AnimalView.cs

Klasa **AnimalView** reprezentuje widok i zarządzanie danymi zwierząt w aplikacji VetClinic. Jest to kontrolka użytkownika odpowiedzialna za wyświetlanie listy zwierząt, ich wyszukiwanie, edycję, usuwanie oraz wyświetlanie powiązanych wizyt.

Po załadowaniu kontrolki, pobiera dane zwierząt, wizyt i właścicieli z bazy i inicjalizuje zakładki z listami zwierząt podzielonymi według typu. Obsługuje dynamiczne filtrowanie zwierząt na podstawie wpisanego tekstu w polu wyszukiwania, aktualizując widok wyników w osobnej zakładce.

Umożliwia dodawanie nowych zwierząt, edytowanie istniejących, a także usuwanie — z dodatkowym potwierdzeniem, jeśli zwierzę ma przypisane wizyty. Po usunięciu danych odświeża widok listy i szczegóły.

Dla wybranego zwierzęcia wyświetla szczegółowe informacje takie jak imię, gatunek, typ, wiek oraz dane właściciela. Udostępnia również nawigację do widoku właściciela poprzez kliknięcie linku.

Dane wizyt przypisanych do wybranego zwierzęcia są ładowane asynchronicznie i wyświetlane w tabeli z informacjami o dacie wizyty, opisie oraz lekarzu prowadzącym.

Kontrolka zarządza również dynamicznym przeładowywaniem listy zwierząt w zakładkach, aby odzwierciedlić zmiany po edycji lub usunięciu. Posiada mechanizmy do obsługi zdarzeń interfejsu użytkownika, takich jak zmiana zaznaczenia, kliknięcia przycisków i zmiany tekstu wyszukiwania.

Całość stanowi kluczowy element interakcji użytkownika z danymi zwierząt w aplikacji, łącząc zarządzanie danymi z wygodnym i responsywnym interfejsem.

DoctorView.cs

Klasa **DoctorView** to kontrolka użytkownika odpowiedzialna za zarządzanie i wyświetlanie danych lekarzy w aplikacji VetClinic. Umożliwia przeglądanie listy lekarzy, szczegółów wybranego lekarza oraz jego wizyt.

Po załadowaniu widoku, klasa asynchronicznie pobiera dane lekarzy, wizyt oraz osób z bazy i inicjalizuje listę lekarzy wraz z danymi wybranego lekarza oraz tabelą jego wizyt. Widok umożliwia również dodawanie nowych lekarzy, edytowanie istniejących danych oraz usuwanie lekarzy, przy czym usunięcie jest możliwe tylko wtedy, gdy lekarz nie ma przypisanych żadnych wizyt.

Szczegóły lekarza wyświetlane w panelu obejmują imię i nazwisko, specjalizację, tryb pracy, telefon, email oraz liczbę odbytych i przyszłych wizyt. Lista wizyt lekarza prezentowana jest w formie tabeli z datami i listą przypisanych leków.

Widok obsługuje zmianę zaznaczenia lekarza na liście, co powoduje aktualizację wyświetlanych danych i tabeli wizyt. Dodatkowo zapewnia możliwość powrotu do głównego panelu danych po operacjach dodawania lub edytowania.

Całość tworzy interfejs do zarządzania informacjami o lekarzach oraz ich harmonogramie wizyt w klinice weterynaryjnej.

MainMenuView.cs

Klasa **MainMenuView** to kontrolka użytkownika pełniąca rolę głównego panelu w aplikacji VetClinic, która wyświetla podsumowanie i statystyki dotyczące wizyt oraz leków.

Po załadowaniu widoku, asynchronicznie ładuje dane leków i wizyt z bazy danych, a następnie prezentuje je w dwóch formatach — w tabeli wizyt zaplanowanych na przyszłość oraz w wykresach. Jeden wykres pokazuje liczbę wizyt pogrupowanych miesięcznie, a drugi przedstawia ilość dostępnych leków, posortowanych według ich ilości.

Metoda **LoadDataToGrid** pobiera z bazy wizyty, które mają datę większą niż aktualna, wraz z powiązanymi danymi lekarza i zwierzęcia, a następnie wyświetla je w tabeli, gdzie kolumny można sortować.

Metody **LoadDataToChart** oraz **LoadMedQuantityChart** generują wykresy — pierwszy pokazuje liczbę wizyt na przestrzeni miesięcy, a drugi liczbę wizyt przypadających na poszczególne leki, wykorzystując wykres słupkowy.

Przycisk „Połącz” umożliwia konfigurację i nawiązanie połączenia z bazą danych na podstawie wprowadzonych parametrów (nazwa użytkownika, hasło, nazwa bazy), a następnie wykonuje inicjalizację danych. W przypadku problemów z połączeniem lub błędów wyświetla odpowiednie komunikaty.

Całość stanowi centralne miejsce, gdzie użytkownik ma szybki dostęp do kluczowych informacji o wizytach i lekach oraz może kontrolować połączenie z bazą danych.

MedicineView.cs

Klasa `MedicineView` jest kontrolką użytkownika odpowiedzialną za zarządzanie danymi leków w systemie kliniki weterynaryjnej. Umożliwia wyświetlanie listy leków, szczegółów wybranego leku, oraz wizyt, na których dany lek był użyty.

Podczas ładowania kontrolka pobiera z bazy danych listy leków i wizyt, które są następnie wykorzystywane do aktualizacji widoków. Udostępnia funkcje do dodawania nowych leków oraz edycji istniejących poprzez otwieranie odpowiednich formularzy. Usuwanie leku jest możliwe tylko wtedy, gdy lek nie był wykorzystany na żadnej wizycie, co zabezpiecza spójność danych.

Klasa obsługuje także wyświetlanie wizyt powiązanych z wybranym lekiem w tabeli, a w przypadku braku takich wizyt pokazuje komunikat informujący o ich braku. Ponadto oferuje możliwość składania zamówień na leki, otwierając dedykowany formularz.

Zmiana wyboru leku na liście powoduje aktualizację szczegółów i powiązanych wizyt, a metoda `refresh` pozwala na ręczne odświeżenie wszystkich widoków, aby zapewnić aktualność danych prezentowanych użytkownikowi.

OrderView.cs

Klasa `OrderView` jest kontrolką użytkownika odpowiedzialną za zarządzanie zamówieniami leków w aplikacji kliniki weterynaryjnej. Po załadowaniu kontrolka pobiera z bazy danych listę zamówień oraz leków i wyświetla je w tabeli.

Umożliwia usuwanie wybranych zamówień, aktualizując po tym listę zamówień i odświeżając widok. Wyświetla również podsumowanie z łączną liczbą zamówień.

Klasa korzysta z kontekstu bazy danych do pobierania i modyfikacji danych oraz synchronizuje dane z główną pamięcią aplikacji (`MainForm`), co pozwala na spójne zarządzanie danymi między różnymi widokami.

OwnerView.cs

Klasa `OwnerView` to kontrolka użytkownika odpowiedzialna za wyświetlanie i zarządzanie właścicielami zwierząt w aplikacji kliniki weterynaryjnej. Po załadowaniu pobiera z bazy

danych listę właścicieli oraz ich zwierząt, wyświetlając te dane w odpowiednich listach i panelach.

Umożliwia dodawanie, edytowanie oraz usuwanie właścicieli. Przy usuwaniu przypisuje zwierzętom właściciela o wartości -1, aby zachować integralność danych. Pozwala także na wyszukiwanie właścicieli po imieniu i nazwisku.

Widok pokazuje szczegółowe informacje o wybranym właścicielu, takie jak imię, nazwisko, wiek, data urodzenia, liczba zarejestrowanych zwierząt, numer telefonu i email. Synchronizuje wybrany właściciel z listą oraz umożliwia przełączanie widoków w panelu edycji danych.

VisitView.cs

Klasa `VisitView` to kontrolka użytkownika odpowiedzialna za wyświetlanie i zarządzanie wizytami w klinice weterynaryjnej. Po załadowaniu pobiera listę wizyt, lekarzy oraz leków z bazy danych i prezentuje je w tabeli.

Pozwala na wyświetlanie szczegółowych informacji o wybranej wizycie, w tym dane zwierzęcia, lekarza, zastosowane leki, datę oraz opis wizyty. Umożliwia edycję i usuwanie wizyt oraz dynamiczne przeładowanie danych.

Przy zmianie wyboru wizyty w tabeli aktualizuje widok szczegółów wizyty. Panel edycji wizyty jest zarządzany poprzez dodanie formularza edycji do kontenera, a panel można przywrócić do widoku podstawowego.

Formularze dodawania

AnimalAddForm.cs

AnimalAddForm to formularz użytkownika służący do dodawania lub edytowania zwierząt w systemie kliniki weterynaryjnej.

Konstruktor umożliwia inicjalizację formularza w trybie dodawania nowego zwierzęcia lub edycji istniejącego na podstawie przekazanych danych.

Metody AddAnimal oraz EditAnimal asynchronicznie dodają lub aktualizują zwierzę w bazie danych i odświeżają lokalną listę zwierząt.

initializeView przygotowuje widok, ustawiając kontrolki do wyszukiwania właściciela oraz listy typów i gatunków zwierząt.

Formularz obsługuje dynamiczne filtrowanie listy właścicieli na podstawie wpisywanego tekstu, umożliwiając wybór właściwego właściciela zwierzęcia z podpowiedzi.

Typy i gatunki zwierząt są ładowane dynamicznie na podstawie globalnego słownika, co pozwala na dopasowanie gatunków do wybranego typu.

W zdarzeniu acceptButton_Click dane z formularza są weryfikowane, a następnie w zależności od trybu formularza tworzony jest nowy wpis lub edytowany istniejący zwierzę, po czym widok główny jest odświeżany, a formularz zamykany.

Przycisk anulowania wycofuje zmiany i powraca do poprzedniego widoku bez zapisywania.

Klasa ta odpowiada za interaktywną obsługę danych zwierząt, zapewniając wygodny interfejs do ich dodawania i modyfikacji wraz z przypisaniem do właściciela.

DoctorAddForm.cs

DoctorAddForm to kontrolka użytkownika służąca do dodawania i edytowania danych lekarzy w systemie kliniki weterynaryjnej.

Konstruktor inicjalizuje formularz w trybie dodawania nowego lekarza lub edycji istniejącego wraz z wczytaniem dostępnych specjalizacji i miejsc pracy.

Metody AddDoctor i EditDoctor asynchronicznie zapisują zmiany w bazie danych i odświeżają lokalną listę lekarzy.

Formularz sprawdza poprawność wprowadzonych danych, w tym unikalność numeru telefonu i adresu e-mail oraz format tych danych, a także w przypadku edycji uniemożliwia zmianę specjalizacji, jeśli lekarz ma przypisane wizyty.

Po zatwierdzeniu zmian dane lekarza są dodawane lub aktualizowane, a widok listy lekarzy i wizyt jest odświeżany.

Przycisk anulowania wycofuje zmiany i wraca do poprzedniego widoku bez zapisywania.

Klasa zapewnia interfejs do zarządzania danymi lekarzy, dbając o ich spójność i integralność w systemie.

MedicineAddForm.cs

MedicineAddForm to kontrolka użytkownika przeznaczona do dodawania i edytowania leków w systemie kliniki weterynaryjnej.

Konstruktor ustawia tryb formularza (dodawanie lub edycja) oraz inicjalizuje pola formularza, wypełniając je danymi w przypadku edycji.

Metody AddMed i EditMed asynchronicznie zapisują nowy lek lub aktualizują istniejący w bazie danych, a następnie odświeżają lokalną listę leków.

Formularz waliduje, czy nazwa leku została podana oraz czy ilość jest w dopuszczalnym zakresie (nie większa niż 100).

Po zatwierdzeniu zmian lek jest dodawany lub modyfikowany, a widok leków jest aktualizowany i następuje powrót do poprzedniego panelu.

Przycisk anulowania powoduje wycofanie zmian i powrót do widoku leków bez zapisywania.

Klasa umożliwia łatwe zarządzanie stanem leków w systemie, dbając o poprawność wprowadzanych danych.

OrderAddForm.cs

OrderAddForm to kontrolka użytkownika odpowiedzialna za składanie zamówień na leki w systemie kliniki weterynaryjnej.

Po inicjalizacji formularza możliwe jest wprowadzenie ilości leku do zamówienia.

Metoda AddOrder asynchronicznie zapisuje nowe zamówienie w bazie danych oraz aktualizuje ilość dostępnego leku, a następnie odświeża lokalne listy zamówień i leków.

Po zatwierdzeniu zamówienia tworzone jest nowe zamówienie z przypisanym lekiem, podaną ilością i aktualną datą.

Formularz automatycznie zwiększa stan magazynowy leku o zamówioną ilość, a następnie zapisuje zmiany i zamyka panel zamówienia.

Przycisk anulowania zamyka panel bez dokonywania zmian.

Klasa ułatwia zarządzanie zamówieniami i stanem leków, dbając o spójność danych w systemie.

OwnerAddForm.cs

Klasa reprezentuje formularz do dodawania i edytowania informacji o właścicielach zwierząt w aplikacji kliniki weterynaryjnej. Umożliwia użytkownikowi wprowadzenie podstawowych danych takich jak imię, nazwisko, data urodzenia, adres email oraz numer telefonu. W trakcie zatwierdzania formularza klasa przeprowadza walidację danych — sprawdza, czy wszystkie pola są wypełnione, czy numer telefonu składa się z 9 cyfr oraz czy adres email ma prawidłowy format. Dodatkowo kontroluje, czy podany email oraz numer telefonu nie są już przypisane do innego właściciela w bazie danych, co zapobiega duplikatom.

Klasa obsługuje dwa tryby działania: dodawanie nowego właściciela oraz edycję danych istniejącego. W trybie dodawania generowany jest nowy unikalny identyfikator, a następnie tworzony jest nowy obiekt reprezentujący właściciela, który zostaje dodany do bazy danych. W trybie edycji aktualizowane są pola istniejącego rekordu właściciela i zapisywane zmiany. Po pomyślnym zapisaniu danych formularz zamyka się, a lista właścicieli jest odświeżana, aby pokazać najnowsze informacje. W przypadku anulowania operacji następuje powrót do widoku listy właścicieli bez wprowadzania zmian.

Dzięki temu formularzowi użytkownik może wygodnie zarządzać danymi właścicieli, a aplikacja zapewnia spójność i poprawność wprowadzanych informacji.

VisitAddForm.cs

Klasa `VisitAddForm` reprezentuje formularz odpowiedzialny za dodawanie oraz edytowanie wizyt pacjentów w systemie kliniki weterynaryjnej. Została zaimplementowana jako kontrolka użytkownika (`UserControl`) i obsługuje dwa tryby działania: dodawanie nowej wizyty lub modyfikację istniejącej. Wybór trybu odbywa się na podstawie wywołanego konstruktora — jeden z nich jest przeznaczony do tworzenia nowych wpisów, drugi do edycji wcześniej zapisanych danych.

W formularzu użytkownik może wybrać zwierzę, lekarza, datę wizyty, dodać opis oraz zaznaczyć leki, które mają być przypisane do wizyty. Lista lekarzy jest filtrowana dynamicznie na podstawie typu zwierzęcia, co pozwala dopasować specjalizację lekarza do konkretnego przypadku. Wybór zwierzęcia odbywa się za pomocą pola tekstowego z funkcją wyszukiwania oraz listy wyników ułatwiającej wybór z dostępnych rekordów.

Dla trybu edycji zaimplementowano ładowanie aktualnych danych dotyczących konkretnej wizyty, w tym przypisanych leków, daty, opisu oraz lekarza. Leki są przedstawione na liście z możliwością zaznaczania wielu pozycji. Wybór lekarza i data wizyty podlegają dodatkowemu sprawdzeniu — system nie pozwala zarejestrować wizyty, jeśli dany lekarz ma już pięć wizyt w wybranym dniu lub jeśli dzień przypada na weekend.

W momencie zatwierdzenia formularza dane są walidowane. Sprawdzana jest kompletność pól, poprawność formatu tekstowego identyfikującego zwierzę, zgodność ze strukturą danych systemu oraz unikalność warunków umożliwiających zapis (np. liczba wizyt lekarza). Jeśli wszystkie warunki są spełnione, tworzony jest nowy rekord wizyty lub modyfikowany istniejący. Zmiany są zapisywane do bazy danych przy użyciu Entity Framework Core, a dane wizyt są synchronizowane z pamięcią główną aplikacji (`MainForm.wizyty`).

Po zapisaniu lub anulowaniu operacji następuje powrót do odpowiedniego widoku: listy wizyt lub widoku wizyt danego zwierzęcia, w zależności od miejsca wywołania formularza. Dzięki temu użytkownik ma intuicyjny dostęp do funkcji zarządzania wizytami w aplikacji, z zapewnieniem integralności i aktualności danych.

Constants.cs

Klasa `Constants` w przestrzeni nazw `VetClinic` jest klasą statyczną przechowującą dane referencyjne oraz pomocnicze metody wykorzystywane w aplikacji weterynaryjnej. Jej celem jest centralizacja informacji wykorzystywanych globalnie, co ułatwia utrzymanie i rozwój projektu.

Pole `AnimalType` zawiera tablicę typów zwierząt, jakie mogą być obsługiwane przez system, takich jak zwierzęta domowe, gospodarskie, egzotyczne ptaki, gady i płazy oraz dzikie zwierzęta.

Pole `Tryby` określa tryby wizyt weterynaryjnych, które są realizowane w trybie klinicznym (stacjonarnym) lub terenowym (wyjazdowym).

Słownik `AnimalSpeciesByType` powiązuje każdy typ zwierzęcia z listą odpowiadających mu gatunków. Na przykład dla typu "Zwierzeta domowe" znajdują się gatunki takie jak "Pies" i "Kot", natomiast dla "Gady i plazy" – "Zolw", "Waz" i inne.

Lista `Diagnoses` zawiera typowe diagnozy stosowane w systemie, takie jak "Zapalenie ucha", "Borelioza", "Niewydolność nerek" czy "Pasożyty wewnętrzne". Są one wykorzystywane w procesie dokumentowania wizyt i leczenia.

Lista `Medicines` to zbiór leków stosowanych w leczeniu różnych schorzeń zwierząt. Zawiera nazwy takie jak "Amoksycylina", "Meloksikam", "Ketoprofen", "Furosemid" oraz wiele innych leków, zarówno przeciwbólowych, przeciwzapalnych, jak i antybiotyków.

Metoda `IsValidEmail` to pomocnicza metoda walidująca poprawność adresu e-mail na podstawie wyrażenia regularnego. Zwraca `true`, jeśli adres ma poprawną składnię.

Pole `CurrentConnectionString` przechowuje bieżący connection string do bazy danych PostgreSQL używanej przez aplikację. Domyślnie wskazuje na lokalną instancję bazy o nazwie `VetClinic`.

Pola `username`, `password` i `name` przechowują dane aktualnie zalogowanego użytkownika. Ponieważ są publiczne i statyczne, nie powinny być używane w środowisku produkcyjnym ze względów bezpieczeństwa.

Metoda `CreateContext` tworzy i zwraca nową instancję kontekstu bazy danych `AppDbContext`, skonfigurowaną z użyciem bieżącego connection stringa oraz z włączonym logowaniem wrażliwych danych. Umożliwia to szybki dostęp do bazy w dowolnym miejscu aplikacji bez konieczności ręcznego konfigurowania opcji połączenia.

Klasa pełni rolę centralnego repozytorium danych i konfiguracji, co jest przydatne w niewielkich projektach. W przypadku większych systemów zaleca się refaktoryzację i podział tej klasy na wyspecjalizowane moduły.

Program.cs

Klasa `Program` zawiera główny punkt wejścia (`Main`) dla aplikacji weterynaryjnej działającej jako aplikacja Windows Forms. Metoda `Main` jest oznaczona atrybutem `[STAThread]`, co jest wymagane przez aplikacje graficzne korzystające z technologii COM, takich jak Windows Forms.

Na początku działania aplikacji tworzona jest instancja fabryki kontekstu bazy danych `AppDbContextFactory`. Przy jej użyciu uzyskiwany jest kontekst bazy danych, który służy do wstępnego zainicjowania danych w systemie za pomocą klasy `DbSeeder`. Metoda `SeedDatabase` wypełnia bazę danych przykładowymi lub wymaganymi danymi, jeżeli są one potrzebne do prawidłowego działania aplikacji. Czynność ta wykonywana jest tylko raz przy uruchomieniu, przed startem interfejsu graficznego.

Po zakończeniu inicjalizacji bazy danych uruchamiana jest konfiguracja środowiska graficznego aplikacji przez `ApplicationConfiguration.Initialize()`. Następnie za pomocą `Application.Run(new MainForm())` uruchamiany jest główny formularz aplikacji – `MainForm`, który stanowi interfejs użytkownika umożliwiający zarządzanie danymi kliniki weterynaryjnej, takimi jak zwierzęta, wizyty, lekarze i leki.

Ta struktura gwarantuje, że baza danych zostanie zainicjalizowana poprawnie jeszcze przed wyświetleniem okna głównego, co pozwala uniknąć potencjalnych błędów wynikających z braku wymaganych danych.