Projekt aplikacji

*Pharmacy*

Wersja v\_23\_05\_23\_1

Data 23.05.2023

Spis treści

[1. Wstęp 3](#_Toc138680191)

[2. Cel i zakres dokumentu 3](#_Toc138680192)

[3. Definicja architektury aplikacji 3](#_Toc138680193)

[4. Cele i ograniczenia architektury 4](#_Toc138680194)

[4.1 Cele architektury 4](#_Toc138680195)

[4.2 Ograniczenia architektury 4](#_Toc138680196)

[5. Obraz logiczny aplikacji 4](#_Toc138680197)

[5.1 Charakterystyka pakietów 4](#_Toc138680198)

[5.2 Diagram klas aplikacji 6](#_Toc138680199)

[5.3 Specyfikacja funkcji i metod aplikacji 6](#_Toc138680200)

[6. Dynamiczny obraz modelowanej aplikacji 6](#_Toc138680201)

[6.1 Diagram sekwencji UML dla obiektów 8](#_Toc138680202)

[6.2 Diagram aktywności UML dla obiektów 9](#_Toc138680203)

[7. Statyczny obraz modelowanej aplikacji 9](#_Toc138680204)

[7.1 Diagram komponentów UML 9](#_Toc138680205)

[7.2 Diagram instalacji UML 10](#_Toc138680206)

[8. Projekt bazy danych 10](#_Toc138680207)

# 1. Wstęp

Niniejszy dokument przedstawia projekt systemu dla aplikacji apteki, której celem jest umożliwienie pracownikom logowania, rejestracji oraz zarządzania antybiotykami i suplementami. Dokument został przygotowany w celu dostarczenia kompleksowego opisu architektury, funkcjonalności i ograniczeń tej aplikacji.

W ramach tego projektu, aplikacja została podzielona na różne moduły, umożliwiające zarządzanie procesami takimi jak logowanie, rejestracja, dodawanie antybiotyków i suplementów oraz przeglądanie dostępnych produktów. Architektura systemu została starannie zaprojektowana, aby zapewnić skalowalność, wydajność i bezpieczeństwo aplikacji.

# 2. Cel i zakres dokumentu

Celem niniejszego dokumentu jest zapewnienie kompleksowego opisu projektu systemu dla aplikacji apteki. Dokument ma na celu dostarczenie szczegółowych informacji dotyczących architektury, funkcjonalności i ograniczeń tej aplikacji, umożliwiając zrozumienie i skuteczne wdrożenie systemu.

Niniejszy dokument obejmuje pełny zakres projektu systemu dla aplikacji apteki. Zawiera opis architektury aplikacji, w tym charakterystykę pakietów, diagramy pakietów UML z specyfikacją interfejsów oraz diagram klas aplikacji. Ponadto, dokument przedstawia specyfikację funkcji i metod aplikacji.

Dokument kończy się projektem bazy danych, który obejmuje diagram encji.

Zakres dokumentu jest szeroki i obejmuje wszystkie istotne aspekty projektu systemu dla aplikacji apteki. Ma on na celu dostarczenie kompletnego i spójnego opisu systemu, umożliwiającego zrozumienie, wdrożenie i dalsze rozwijanie aplikacji.

# 3. Definicja architektury aplikacji

Definicja architektury aplikacji jest kluczowym elementem projektu systemu dla aplikacji apteki. Ten rozdział dostarcza ogólnego opisu struktury i zasad, które zostały przyjęte podczas projektowania architektury aplikacji.

* Warstwy architektury:

Architektura aplikacji apteki została zaprojektowana z wykorzystaniem wzorca warstwowego, który umożliwia logiczne podzielenie aplikacji na warstwy, z każdą z nich pełniącą określoną rolę i funkcję. Główne warstwy architektury to:

* + Warstwa interfejsu użytkownika (UI) - Warstwa UI odpowiada za prezentację danych użytkownikowi i interakcję z nim.
  + Warstwa logiki biznesowej - Warstwa logiki biznesowej jest odpowiedzialna za przetwarzanie danych, logikę operacji oraz obsługę reguł biznesowych związanych z zarządzaniem antybiotykami i suplementami. Zawiera również logikę autoryzacji użytkowników.
  + Warstwa dostępu do danych - Warstwa dostępu do danych jest odpowiedzialna za komunikację z bazą danych.
* Komunikacja między warstwami:

Komunikacja między warstwami odbywa się zgodnie z zasadami architektury wielowarstwowej. Warstwa interfejsu użytkownika komunikuje się z warstwą logiki biznesowej za pomocą odpowiednich interfejsów i wywołań funkcji. Warstwa logiki biznesowej korzysta z usług warstwy dostępu do danych w celu pobierania i zapisywania danych.

* Bezpieczeństwo aplikacji:

Bezpieczeństwo aplikacji apteki jest priorytetem. Aby zapewnić odpowiednią ochronę danych i funkcji aplikacji, zastosowane zostały odpowiednie mechanizmy zabezpieczeń, takie jak autoryzacja użytkowników.

# 4. Cele i ograniczenia architektury

4.1 Cele architektury

Architektura aplikacji apteki została zaprojektowana w celu osiągnięcia następujących celów:

* Zapewnienie łatwości użytkowania: Aplikacja powinna być intuicyjna i przyjazna dla użytkowników, umożliwiając im wygodne i efektywne korzystanie z jej funkcjonalności. Interfejs użytkownika powinien być przejrzysty, estetyczny i ergonomiczny.
* Bezpieczeństwo danych: Aplikacja powinna zapewniać wysoki poziom bezpieczeństwa danych użytkowników, zarówno pod względem dostępu, jak i przechowywania.

4.2 Ograniczenia architektury

Podczas projektowania architektury aplikacji apteki uwzględnione zostały następujące ograniczenia:

* Dostępność technologii: Architektura aplikacji musi być zgodna z dostępnymi technologiami i narzędziami, które są dostępne dla zespołu projektowego. Konieczne jest uwzględnienie ograniczeń technologicznych i wybór odpowiednich rozwiązań, które są dostępne i wspierane.
* Budżet i zasoby: Projektowanie architektury aplikacji musi uwzględniać dostępne zasoby finansowe i ludzkie. Ograniczenia budżetowe mogą wpływać na wybór technologii, narzędzi i infrastruktury, a ograniczenia zasobów ludzkich mogą wpływać na czas i wysiłek potrzebny do implementacji architektury.
* Wymagania regulacyjne i zgodność: Architektura aplikacji powinna być zgodna z obowiązującymi przepisami i regulacjami dotyczącymi ochrony danych osobowych, bezpieczeństwa informacji oraz innych wymogów prawnych. Konieczne jest uwzględnienie tych ograniczeń i dostosowanie architektury, aby spełniała wszystkie wymagania regulacyjne.

# 5. Obraz logiczny aplikacji

## 5.1 Charakterystyka pakietów

1. Pakiet „Use Cases”:

Opis: Pakiet zawiera listę Use Case’ów zawartych w systemie

1. Pakiet "Wspólne zachowania":

Opis: Zawiera funkcjonalności i komponenty, które są używane przez różne części aplikacji apteki. Obejmuje to wspólne operacje, narzędzia i funkcje, które są wykorzystywane w wielu miejscach w systemie np. logowanie.

1. Pakiet "Klasy":

Opis: zawiera klasy reprezentujące podstawowe obiekty i modele danych używane w aplikacji apteki. Te klasy są wykorzystywane do przechowywania informacji, przetwarzania danych i wykonywania operacji związanych z działaniem systemu.

1. Pakiet "Interfejsy":

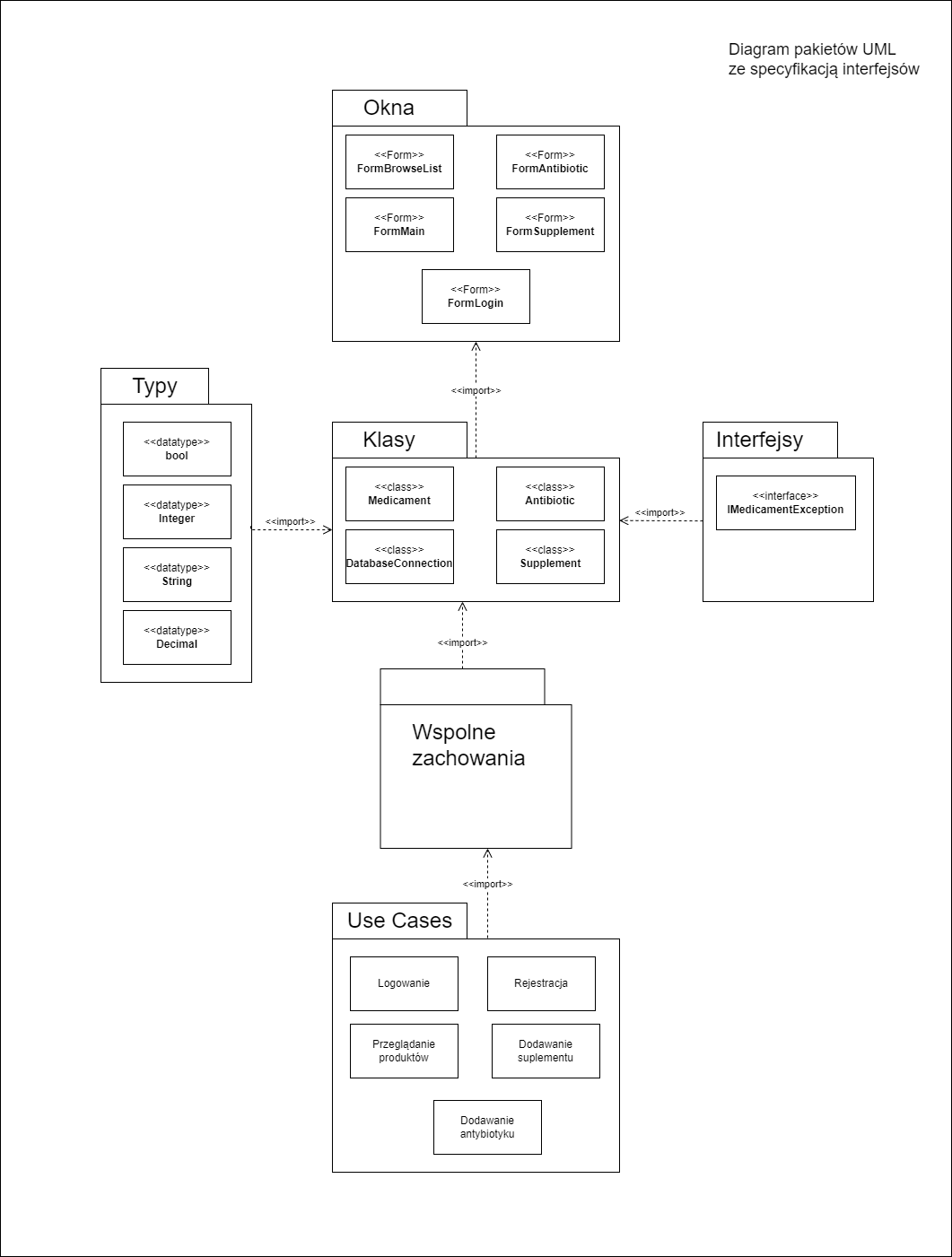
Opis: Zawiera deklaracje interfejsów, które określają zestaw metod i funkcji, które muszą być zaimplementowane przez klasy w celu zapewnienia określonej funkcjonalności w aplikacji apteki.

1. Pakiet „Typy”:

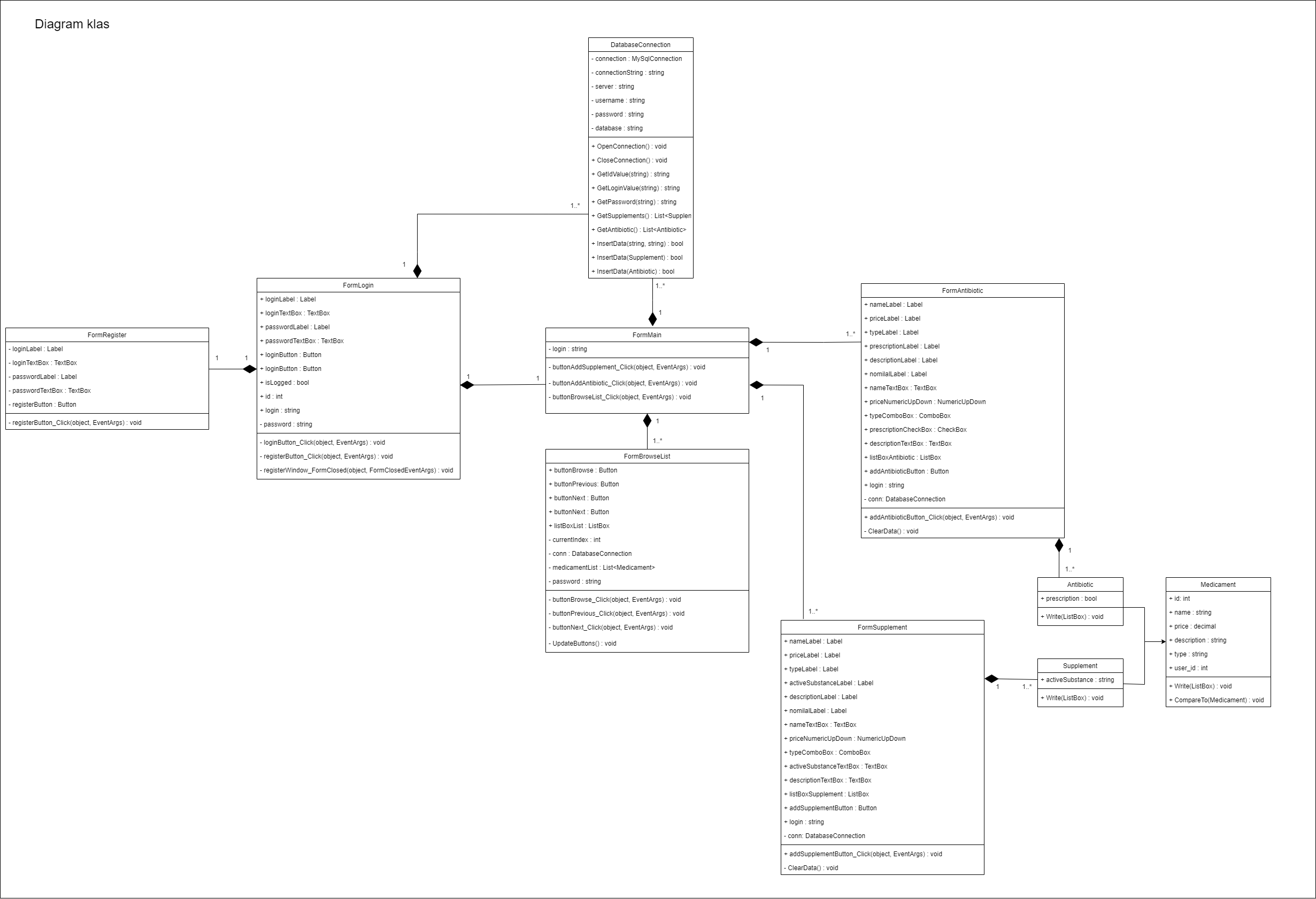
Opis: Zawiera deklaracje różnych typów danych i struktur, które są wykorzystywane w aplikacji apteki do przechowywania i manipulacji danymi.

1. Pakiet „Okna”

Opis: Pakiet "Okna" zawiera klasy i komponenty związane z interfejsem użytkownika w aplikacji apteki. Obejmuje okna, widoki i komponenty graficzne, które umożliwiają użytkownikowi interakcję z systemem.

**

## 5.2 Diagram klas aplikacji



## 5.3 Specyfikacja funkcji i metod aplikacji

**Antibiotic.cs:**

* void Write(ListBox) – dodanie elementów do ListBox w celu wyświetlenia ich.

**DatabaseConnection.cs:**

* void OpenConnection() – Otwarcie połączenia z bazą danych
* void CloseConnection() – Zamknięcie połączenia z bazą danych
* string GetIdValue(string) – Pobranie wartości Id z bazy danych
* string GetLoginValue(string) – Pobranie wartości Login z bazy danych
* string GetPasswordValue(string) – Pobranie wartości Password z bazy danych
* List<Supplement> GetSupplements() – Pobranie listy suplementów z bazy danych
* List<Antibiotic> GetAntibiotic() – Pobranie listy antybiotyków z bazy danych
* bool InsertData(string, string) – Dodanie użytkownika do bazy danych
* bool InsertData(Supplement) – Dodanie suplementu do bazy danych
* bool InsertData(Antibiotic) - Dodanie antybiotyku do bazy danych

**FormBrowseList.cs:**

* void buttonBrowse\_Click(object, EventArgs) – Pobranie listy z bazy danych i wyświetlenie pierwszego obiektu z listy
* void buttonPrevious\_Click(object, EventArgs) – Wyświetlenie poprzednigo obiektu z listy
* void buttonNext\_Click(object, EventArgs) – Wyświetlenie następnego obiektu z listy
* void UpdateButtons() – Uaktualnienie stanu przycisków

**Medicament.cs:**

* virtual void Write(ListBox) – Dodanie obiektu do listy

**FormSupplement.cs:**

* void addSupplementButton\_Click(object, EventArgs) - metoda odpowiadająca za pobranie danych z pól w FormSupplement, za sprawdzenie poprawności wprowadzonych danych oraz za wysłanie prośby o dodanie Suplementu
* void ClearData() - odpowiada za wyczyszczenie pól w FormSupplement

**FormAntibiotic.cs:**

* void addAntibioticButton\_Click(object, EventArgs) - metoda odpowiadająca za pobranie danych z pól w FormAntibiotic, za sprawdzenie poprawności wprowadzonych danych oraz za wysłanie prośby o dodanie Antybiotyku
* void ClearData() - odpowiada za wyczyszczenie pól w FormAntibiotic

**FormRegister.cs:**

* void registerButton\_Click(object, EventArgs) - metoda odpowiadająca za pobranie danych z pól w FormRegister, za sprawdzenie poprawności wprowadzonych danych oraz za wysłanie prośby o dodanie użytkownika

**FormMain.cs:**

* void buttonAddSupplement\_Click(object, EventArgs) - metoda odpowiadająca za wyświetlenie FormSupplement
* void buttonAddAntibiotic\_Click(object, EventArgs) - metoda odpowiadająca za wyświetlenie FormAntibiotic
* void buttonBrowseList\_Click(object, EventArgs) - metoda odpowiadająca za wyświetlenie FormBrowseList

**Supplement.cs:**

* void Write(ListBox) – dodanie elementów do ListBox w celu wyświetlenia ich.

**FormLogin.cs:**

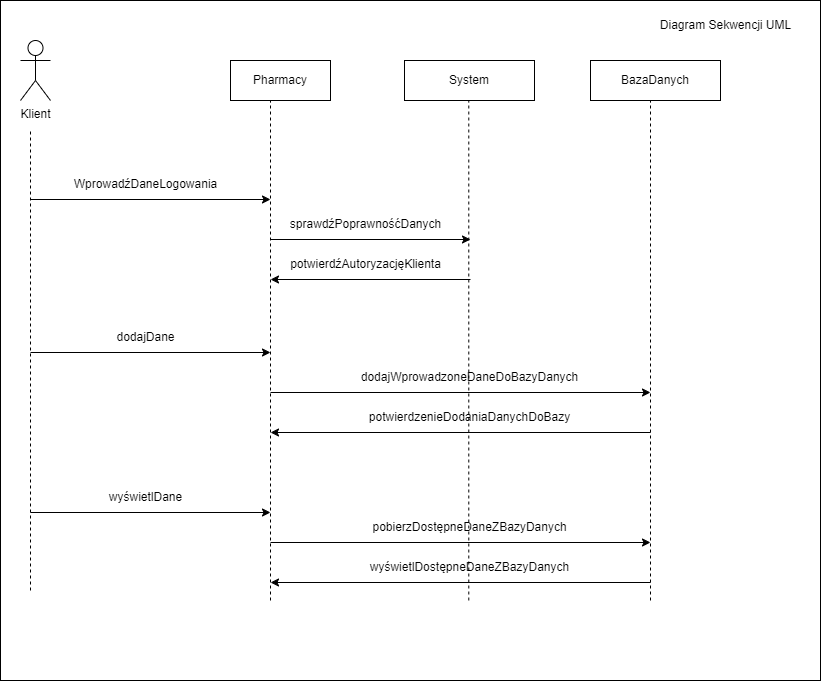
* void loginButton\_Click(object, EventArgs) - metoda odpowiadająca za pobranie danych z pól w FormLogin, za sprawdzenie poprawności wprowadzonych danych oraz za wysłanie prośby o sprawdzenie czy dany użytkownik znajduje się w bazie
* void registerButton\_Click(object, EventArgs) - metoda odpowiadająca za wyświetlenie FormRegister
* void registerWindow\_FormClosed(object, FormClosedEventArgs) - metoda odpowiadająca za sprawdzenie czy Formatka RegisterForm została zamknięta

**Program.cs:**

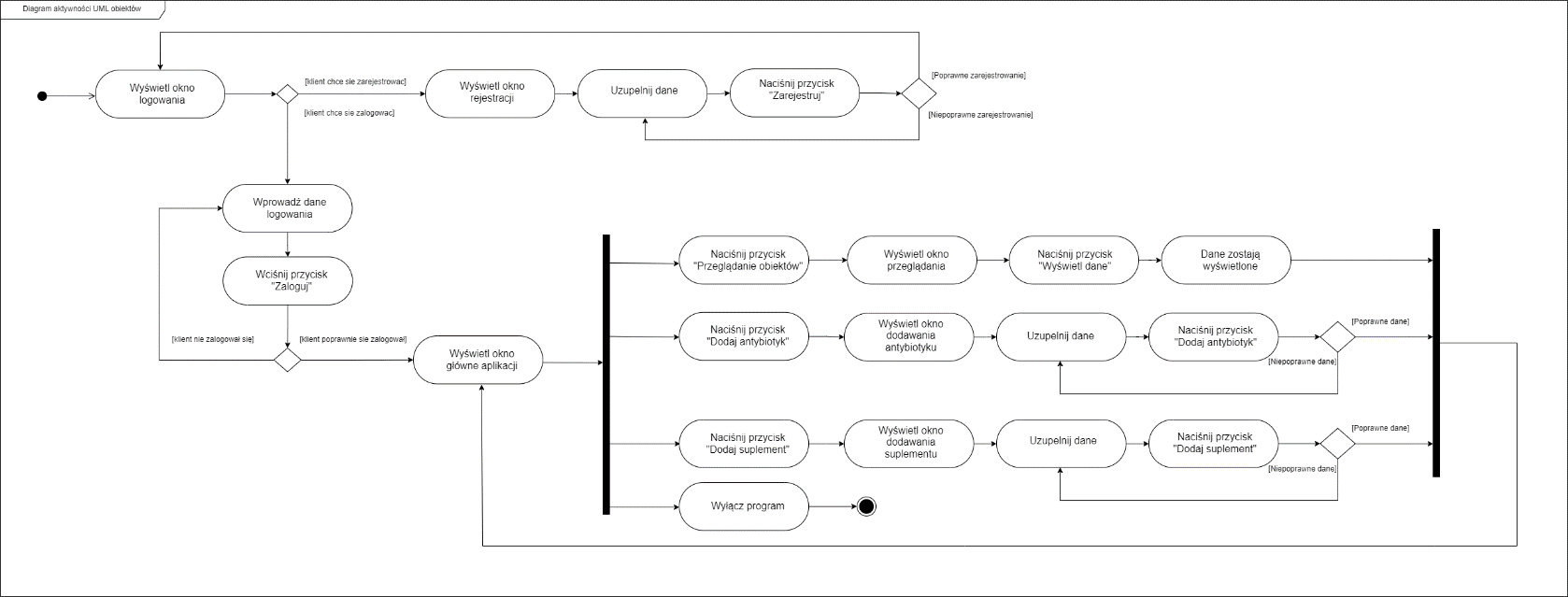
* void Main( ) – Uruchomienie aplikacji

# 6. Dynamiczny obraz modelowanej aplikacji

## 6.1 Diagram sekwencji UML dla obiektów

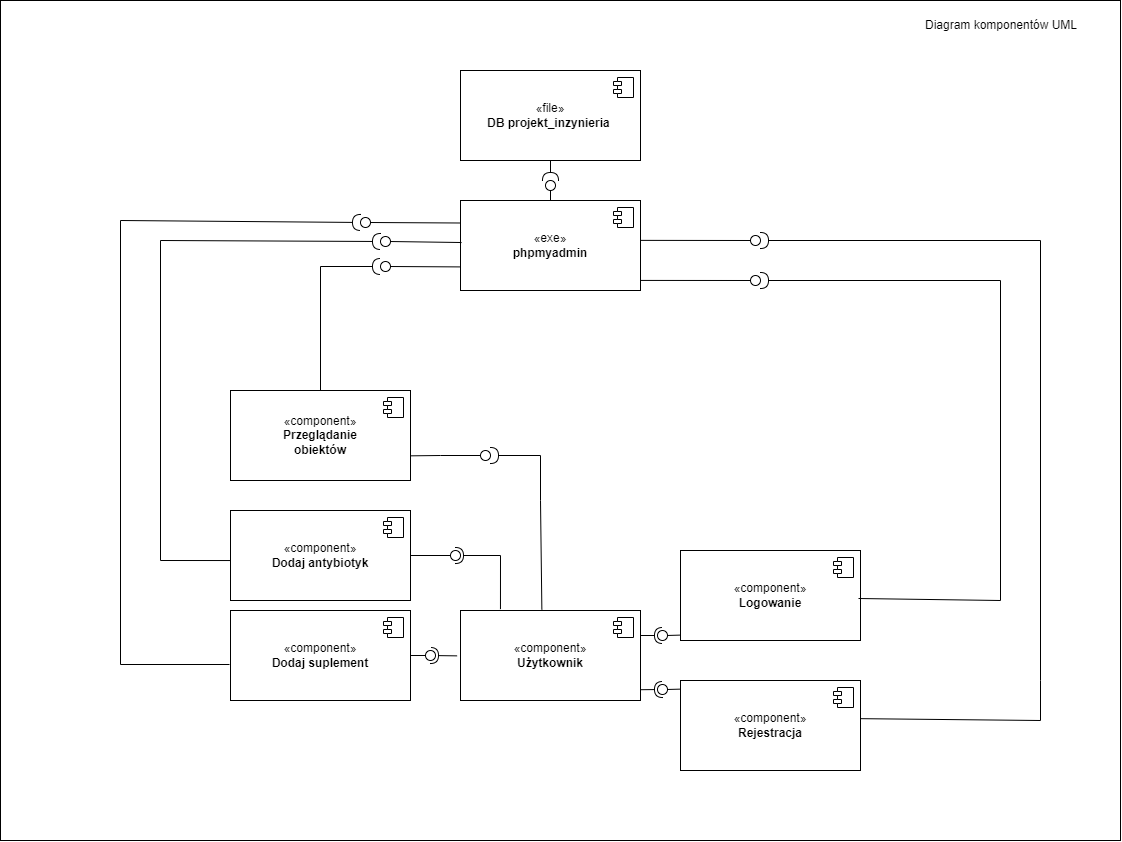
**

## 6.2 Diagram aktywności UML dla obiektów

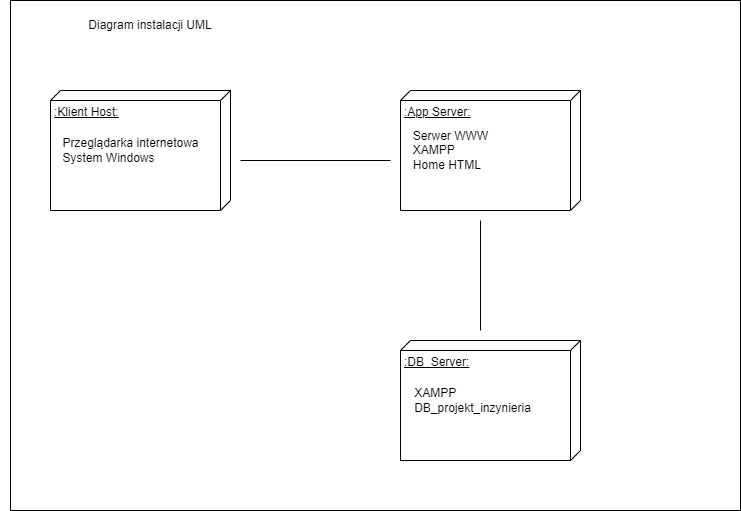


# 7. Statyczny obraz modelowanej aplikacji

## 7.1 Diagram komponentów UML

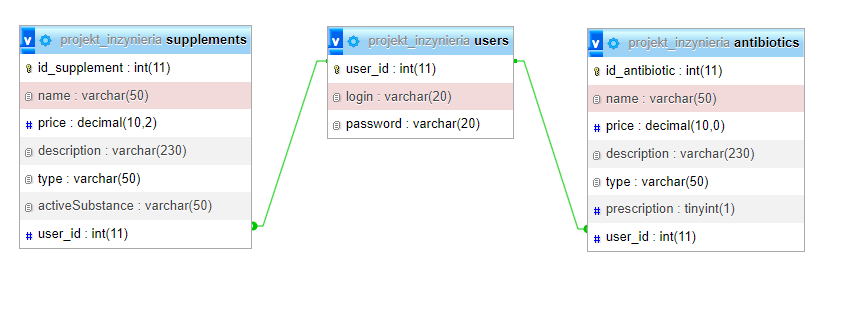


## 7.2 Diagram instalacji UML



# 8. Projekt bazy danych

* Diagram encji:



* Procedury wykonywane na danych:
  + Dodawanie danych do bazy
  + Pobieranie danych z bazy
* Specyfikacja bazy danych "projekt\_inzynieria":

1. Tabela: users

* user\_id (integer) - klucz główny (primary key)
* login (varchar) - nazwa użytkownika
* password (varchar) - hasło użytkownika

2. Tabela: supplements

* id\_supplement (integer) - klucz główny (primary key)
* name (varchar) - nazwa suplementu
* price (decimal) - cena suplementu
* description (varchar) - opis suplementu
* type (varchar) - typ suplementu
* activeSubstance (varchar) - substancja aktywna
* user\_id (integer) - klucz obcy (foreign key) odnoszący się do użytkownika w tabeli users

3. Tabela: antibiotics

* id\_antibiotic (integer) - klucz główny (primary key)
* name (varchar) - nazwa antybiotyku
* price (decimal) - cena antybiotyku
* description (varchar) - opis antybiotyku
* type (varchar) - typ antybiotyku
* prescription (boolean) - czy wymagany jest recepta
* user\_id (integer) - klucz obcy (foreign key) odnoszący się do użytkownika w tabeli users

4. Uwagi:

Baza danych "projekt\_inzynieria" zawiera trzy tabele: users, supplements i antibiotics.

Tabela users przechowuje informacje o użytkownikach, takie jak identyfikator, login i hasło.

Tabela supplements przechowuje informacje o suplementach, takie jak identyfikator, nazwa, cena, opis, typ, substancja aktywna oraz odwołanie do użytkownika za pomocą klucza obcego user\_id.

Tabela antibiotics przechowuje informacje o antybiotykach, takie jak identyfikator, nazwa, cena, opis, typ, informację czy wymagana jest recepta oraz odwołanie do użytkownika za pomocą klucza obcego user\_id.