



# **Państwowa Wyższa Szkoła Informatyki i Przedsiębiorczości w Łomży**

**Wydział Informatyki i Nauk o Żywności**

**Kierunek studiów: Informatyka I stopień**

**Specjalność: Systemy oprogramowania**

**Karol Budzyński**

8096

**ISP Manager – desktopowa aplikacja bazodanowa  
wspierająca pracę dostawcy usług internetowych.**

**ISP Manager - a desktop database application supporting the  
work of an internet service provider.**

**Promotor:**

dr inż. Eugenia Busłowska

.....  
(podpis promotora)

**Łomża 2019**



# SPIS TREŚCI

<b>1. Wstęp .....</b>	<b>4</b>
1.1. Uzasadnienie wyboru tematu pracy .....	4
1.2. Cel i zakres pracy .....	5
<b>2. Technologie wykorzystane w tworzeniu aplikacji .....</b>	<b>7</b>
<b>3. Baza danych .....</b>	<b>9</b>
3.1. Podstawowe informacje o bazie danych .....	9
3.2. Połączenie z bazą danych .....	9
3.3. Struktura bazy danych .....	10
<b>4. Funkcjonalność aplikacji .....</b>	<b>13</b>
4.1. Logowanie i okno główne .....	13
4.2. Menu paneli, przeglądanie i wyszukiwanie danych .....	16
4.3. Panel Klienci .....	18
4.4. Panel Umowy .....	19
4.5. Panel Abonamenty .....	20
4.6. Panel Finanse .....	22
4.7. Panel Zlecenia .....	25
4.8. Panel Urządzenia .....	26
4.9. Panel Pracownicy .....	28
4.10. Zmiana hasła .....	29
<b>5. Podsumowanie i wnioski .....</b>	<b>31</b>
<b>Literatura .....</b>	<b>33</b>
<b>Spis rysunków .....</b>	<b>34</b>

# 1. Wstęp

W dzisiejszych czasach większość z nas posiada stały dostęp do Internetu. Wykorzystujemy go do różnych celów, takich jak zdobywanie informacji, kontakt z przyjaciółmi i rodziną, zdalna kontrola różnorodnych urządzeń. Dostęp do Internetu zapewnia nam różnego rodzaju rozrywki. W związku z tym na rynku panuje coraz większa konkurencja wśród dostawców usług internetowych ISP (*ang. Internet Service Provider*). Są to firmy dostarczające Internet dla użytkowników domowych, przedsiębiorstw oraz organizacji. W tym celu wykorzystują infrastrukturę dostawców telekomunikacyjnych lub infrastrukturę własną [1]. Wśród tych firm możemy wyróżnić dużych operatorów ogólnokrajowych (m.in. Orange, Netia, Plus) oraz małych i średnich operatorów regionalnych bądź lokalnych. Do zarządzania bazą klientów, dostarczanych usług, gromadzenia informacji o płatnościach, umowach, awariach oraz posiadanym sprzęcie sieciowym, dostawcy usług internetowych wykorzystują oprogramowanie bazodanowe, które pozwala na przechowywanie tego typu danych.

Praca podzielona jest na pięć rozdziałów zasadniczych. W rozdziale drugim przedstawiono technologie wykorzystane do stworzenia aplikacji. Rozdział trzeci opisuje bazę danych wykorzystywaną w programie do przechowywania informacji potrzebnych dla dostawcy usług internetowych. Rozdział czwarty został w całości poświęcony funkcjonalności aplikacji. Zostaną w nim opisane wszystkie moduły i możliwości programu. W rozdziale piątym podsumowano wykonaną pracę i zamieszczono wnioski. Na końcu przedstawiono spis literatury i innych źródeł, wykorzystanych w trakcie tworzenia pracy dyplomowej oraz wykaz rysunków.

## 1.1. Uzasadnienie wyboru tematu pracy

Głównym powodem stworzenia aplikacji wspomagającej pracę dostawcy usług internetowych jest mała liczba oprogramowania, które jest skierowane bezpośrednio dla firm dostarczających dostęp do Internetu. Najpopularniejsze rozwiązania dostępne na rynku to: Pyxis4SQL oraz LMS (LAN Management System). Producent podaje, że Pyxis4SQL to oprogramowanie dla ISP (aplikacja bazodanowa) pracujące w systemie Windows w połączeniu z darmową bazą MySQL, służące do zbierania i przetwarzania informacji potrzebnych do

zarządzania rozliczeniami z klientami i sprzętem w małej i średniej firmie świadczącej dostęp do Internetu [2]. Minusem tego rozwiązania jest jego złożoność oraz cena. Jest to bardzo rozbudowany program przeznaczony dla dużych sieci. Natomiast oprogramowanie LMS jest zintegrowanym systemem zarządzania sieciami przeznaczonym dla różnej wielkości dostawców Internetu (ISP). Jest to darmowe rozwiązanie oparte na licencji GNU GPL v2 [3]. Minusem tego rozwiązania jest konieczność posiadania serwera WWW z interpreterem języka PHP. Dodatkowo od użytkowników wymagana jest znajomość systemów UNIX w celu instalacji i konfiguracji tego rozwiązania. Biorąc pod uwagę te braki i niedogodności postanowiłem stworzyć oprogramowanie dla niewielkich dostawców usług internetowych, które będzie proste w obsłudze, będzie posiadało tylko niezbędne funkcje i rozwiązania. Cała aplikacja została wykonana od podstaw. Do jej budowy nie użyto żadnych gotowych systemów ani rozwiązań.

## **1.2. Cel i zakres pracy**

Celem pracy jest zaprojektowanie aplikacji bazodanowej do przechowywania podstawowych danych dotyczących klientów dostawcy usług internetowych oraz jego implementacja w obiektowym języku programowania, jakim jest C#. Stworzenie tego typu oprogramowania ma na celu ułatwienie pracy dostawcy usług internetowych, który poprzez prosty i przejrzysty interfejs będzie miał dostęp do wszystkich potrzebnych informacji na temat klientów.

Aplikacja ma za zadanie spełniać określone założenia funkcjonalne, wśród których najważniejsze to:

- możliwość dodawania klientów, umów zawartych z klientami, pakietów internetowych (abonamentów),
- możliwość naliczania opłat abonamentowych oraz zapisywanie wpłat dokonywanych przez klientów,
- możliwość wyświetlania szczegółowego salda klienta,
- możliwość dodawania zleceń serwisowych i prowadzenie historii przeprowadzonych prac,
- możliwość prowadzenia rejestru urządzeń sieciowych pracujących w sieci,
- możliwość dodawania pracowników (użytkowników aplikacji),

- możliwość przeglądania, edycji i usuwania wszelkich informacji zawartych w bazie danych,
- szybki dostęp do informacji za pomocą wyszukiwarki.

## 2. Technologie wykorzystane w tworzeniu aplikacji

Do stworzenia aplikacji bazodanowej wykorzystano środowisko programistyczne Microsoft Visual Studio 2010, ponieważ jest to zestaw narzędzi, który automatyzuje wiele przyziemnych zadań wymaganych przy tworzeniu aplikacji [4]. Visual Studio posiada dużo wstępnie zdefiniowanych elementów sterujących, w tym szkielet kodu, co pozwala uniknąć konieczności pisania własnego kodu w celu wykonywania powtarzających się zadań. Edytor Visual Studio optymalizuje jakość kodowania. Znaczna część kodu jest oznaczana różnymi kolorami. Narzędzie *IntelliSense* zapewnia wskazówki, które pojawiają się podczas pisania kodu. Dzięki różnym skrótom klawiaturowym wykonujemy wiele zadań. Istnieje kilka refaktoryzacji, które pomagają szybko usprawnić organizację kodu podczas programowania. Na przykład refaktoryzacja *Rename*, pozwala zmienić nazwę identyfikatora w miejscu, w którym jest zdefiniowany, co powoduje zmianę tej nazwy w każdym miejscu w kodzie gdzie jest odwołanie do tego identyfikatora.

Językiem programowania wykorzystanym do stworzenia aplikacji jest C# (*C Sharp*). Został on zaprojektowany i rozwijany przez grupę programistów Microsoft pod kierunkiem Andersa Hejlsberga, który wcześniej był odpowiedzialny za kompilator Turbo Pascala, środowisko Delphi i bibliotekę VCL w firmie Borland [5]. C# to nowoczesny język programowania, ciągle rozwijany przez Microsoft. Obecnie znajduje się w czołówce najpopularniejszych języków programowania na świecie. Wykorzystując C# możemy tworzyć aplikacje na Androida, aplikacje na systemy Windows jak Windows Forms i WPF, aplikacje na iOS. C# służy także do tworzenia stron internetowych. Głównymi cechami, które wpłynęły na wybór C# są:

- wysoka wydajność,
- integracja z bazami danych,
- możliwość łatwego przenoszenia kodu,
- ilość dostępnych bibliotek, zintegrowanych w tym języku,
- wsparcie techniczne i dostęp do kodów źródłowych,
- dostęp do wielu materiałów ułatwiających naukę języka C#

Środowisko bazodanowe, które wybrano do stworzenia aplikacji to MS SQL, gdyż jest to najbardziej ceniona i najpopularniejsza technologia wśród developerów, a także ekspertów bazodanowych. W aplikacji został wykorzystana bezpłatna wersja Express, idealna do nauki i tworzenia aplikacji klasycznych oraz niewielkich (do 10 GB) aplikacji serwerowych działających w oparciu o bazy danych [10]. Technologia ta cechuje się dużą wydajnością oraz integracją z obiektywnym językiem C#, w którym wbudowana jest obsługa MS SQL. Rozwiązanie to cieszy się bardzo dużą popularnością i jest łatwo przyswajalne dla początkujących użytkowników.

Bazy danych są wykorzystywane do przechowywania informacji, np. dane klientów firmy lub spis pracowników. Stosowanie baz danych ułatwia pracę, ponieważ umożliwia szybki dostęp do informacji, pozwala na sortowanie danych według konkretnych wartości, a także wyszukiwanie interesujących nas rekordów. Tabele w bazie zawierają pola i rekordy. Każde pole ma określony typ przechowywania danych oraz długość ciągu, który będzie zachowany w bazie. Tabele można łączyć za pomocą kluczy, tworząc relacje. Istnieją różne typy relacji np.: jeden do jednego lub jeden do wielu [8].

Narzędziem służącym do zarządzania bazą danych MS SQL, które zostało wykorzystane w projekcie to wbudowane w pakiet Microsoft Visual Studio 2010 narzędzie Server Explorer. Pozwala ono na tworzenie i edycję baz danych, zarządzanie tabelami i edycję rekordów oraz tworzenie relacji. Posiada łatwy i przejrzysty interfejs oraz nie wymaga od użytkownika biegłej znajomości języka SQL.



## 3. Baza danych

### 3.1. Podstawowe informacje o bazie danych

W aplikacji użyto bazy danych MS SQL Express, ponieważ jej integracja z językiem C# jest bardzo prosta i intuicyjna. MS SQL Express dzięki swojej szybkości i niezawodności jest doskonałym środowiskiem do tworzenia projektów małej skali. Przy wyborze technologii bazodanowej kierowano się przede wszystkim poziomem jej niezawodności i możliwościami, jakie oferuje baza danych. Zarządzanie bazą odbywa się bezpośrednio z Microsoft Visual Studio 2010, bądź też przez Microsoft SQL Management Studio Express [10]. Dzięki tym programom można na bieżąco kontrolować dane zawarte w bazie.

W bazie danych będą przechowywane informacje o klientach, ich umowach i płatnościach. Baza będzie przechowywać także dane pracowników, informacje o zaplanowanych pracach oraz listę sprzętu sieciowego.

### 3.2. Połączenie z bazą danych

Podstawową funkcją, którą ma wykonywać aplikacja to łączenie z bazą danych MS SQL. Połączenie z bazą zostało zaimplementowane w klasie *SetConnection*, wykorzystując wbudowaną bibliotekę C# *System.Data.SqlClient* [6]. Dopiero po pomyślnym połączeniu z bazą danych, możemy kierować do niej różne zapytania. W tym celu wykorzystano następujący kod:

```
class setConnection
{
    public static SqlConnection con = null;
    public static string conn_str =
        ISP_Manager.Properties.Settings.Default.ISPdbConn
        ectionString;

    public static void Connection()
    {
        con = new SqlConnection(conn_str);
        con.Open();
    }
}
```

Połączenie z bazą odbywa się za pomocą klasy *SetConnection*, w której pobierane są parametry połączenia z naszą bazą danych. Parametry te zapisane są we

właściwościach aplikacji i przekazywane do zmiennej *conn\_str*. Zmienna ta pobiera z ustawień programu parametr *ISPdbConnectionString*, który przechowuje ścieżkę do naszej bazy oraz właściwości połączenia. Jego wartość wygląda następująco:

```
Data Source=.\SQLEXPRESS;  
AttachDbFilename=|DataDirectory|\ISPdb.mdf;  
Integrated Security=True; UserInstance=True;
```

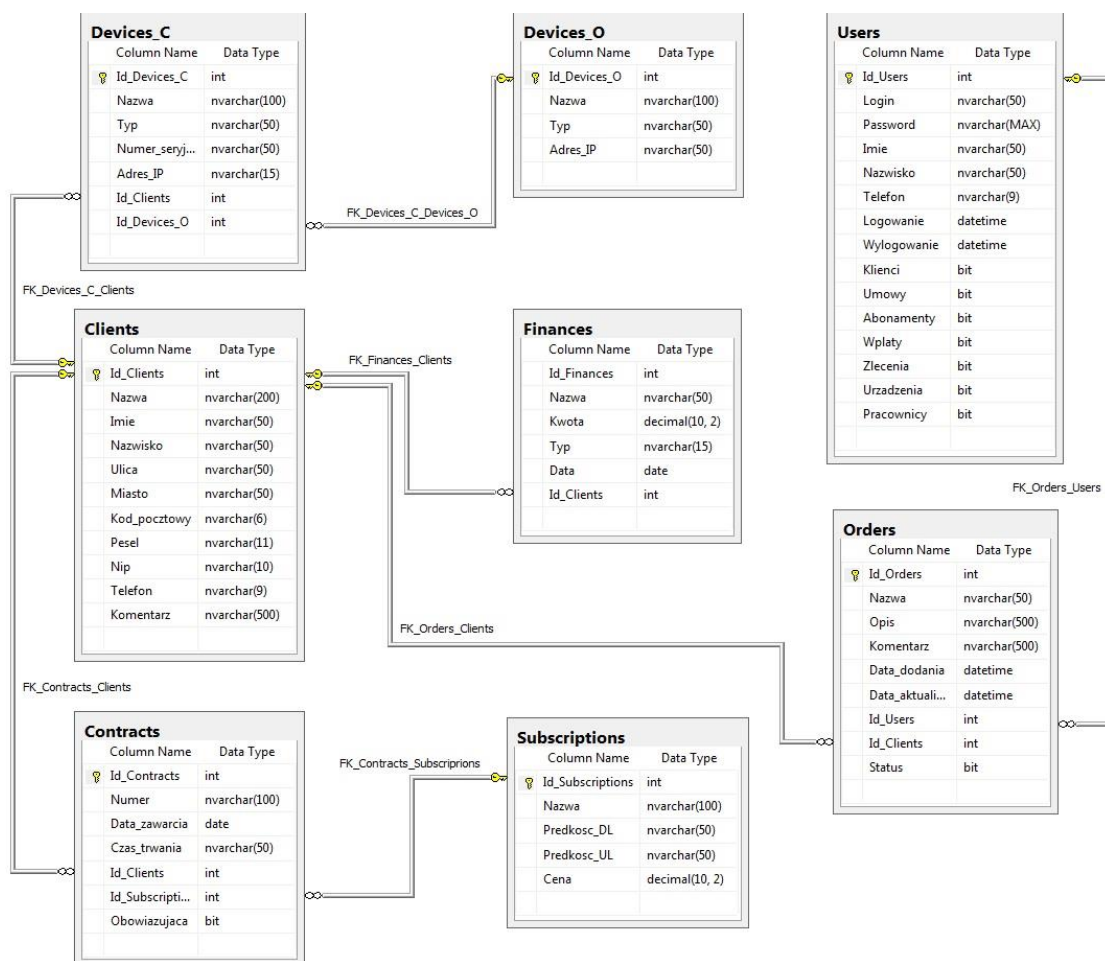
W momencie logowania do aplikacji następuje próba połączenia z bazą danych. Poniższy kod przedstawia jak za pośrednictwem funkcji *try/catch* [9] weryfikowane jest czy udało się ustanowić połączenie:

```
try  
{  
    connection.Open();  
}  
catch (SqlException err)  
{  
    err2 = err.ToString();  
    MessageBox.Show("Błąd połączenia z bazą danych",  
        "Komunikat", MessageBoxButtons.OK,  
        MessageBoxIcon.Warning);  
}
```

Gdy nie udaje się nawiązać połączenia z bazą danych, wyświetlony zostaje komunikat o błędzie oraz kod błędu zostaje zapisany do zmiennej *err2*.

### 3.2. Struktura bazy danych

Utworzona baza danych, zawiera osiem tabel. Powiązane są one ze sobą relacjami [8]. Struktura bazy danych została znormalizowana, tabele są w trzeciej postaci normalnej (3PN) [7]. Wynika to z tego, iż dane przechowywane w tabelach są atomowe, każda z tabel posiada klucz główny i przechowuje dane odnoszące się tylko do danej tabeli. W bazie nie ma zależności przechodnich. Dane w tabelach nie powtarzają się oraz przy tworzeniu tabel uniknięto zbędnych pól. Baza zaprojektowana została z myślą o optymalnym działaniu, co wpłynie na szybkość i poprawność jej działania. Strukturę bazy danych przygotowanej do projektowanej aplikacji ilustruje Rys. 3.1.



**Rys. 3.1.** Struktura bazy danych aplikacji

Przedstawiona struktura bazy danych ukazuje tabele, które powiązane są kluczami obcymi [7]. Tabele zawierają różne pola z określonymi typami danych. Dodatkowo część pól ma przypisaną informację o długości ciągu znaków, jaki może być przechowywany w tabeli. Na przykład identyfikator klienta *Id\_Clients* jest typu *integer* bez określonej długości znaków. Czyli wspomniane pole może przyjąć tylko wartości liczb całkowitych [6]. Natomiast *Kod\_pocztowy* jest typu *nvarchar* o długości znaków, ponieważ standardowy kod pocztowy w Polsce składa się z pięciu cyfr i myślnika [6].

Tabela *Clients*, przechowująca informacje o klientach, jest powiązana z tabelami *Contracts*, *Devices\_C*, *Finances*, *Orders* relacją jeden do wielu, ponieważ jeden klient może mieć zawartych wiele umów, posiadać wiele urządzeń w sieci, dokonywać wielu operacji finansowych oraz zgłaszać wiele zleceń serwisowych. Tabela *Subscriptions*, w której są dane o dostępnych pakietach internetowych, jest w relacji jeden do wielu do tabeli *Contracts*, przechowującej

informacje na temat zawartych umów, gdyż jeden abonament może występować na wielu umowach. Tabela *Devices\_O*, zawierająca dane urządzeń sieciowych dostawcy usług internetowych, łączy się z tabelą *Devices\_C*, gdzie zawarte są informacje o klienckich urządzeniach sieciowych, relacją jeden do wielu, ponieważ do jednego urządzenia operatora jest podłączonych wiele urządzeń klienckich. Tabela *Users*, która przechowuje dane o użytkownikach aplikacji oraz pracownikach, jest w relacji jeden do wielu z tabelą *Orders*, w której zawarte są informacje o zgłoszeniach serwisowych, z tego względu, iż jeden pracownik może być oddelegowany do wielu zleceń.

## 4. Funkcjonalność aplikacji

### 4.1. Logowanie i okno główne

Dostęp do aplikacji bazodanowych zawierających poufne dane musi odbywać się poprzez autoryzację użytkownika. Autoryzacja następuje po uruchomieniu programu i wpisaniu danych autoryzujących. Panel logowania jest pierwszą planszą, którą ujrzy użytkownik po uruchomieniu aplikacji. Przedstawione zostało ono na Rys. 4.1.



**Rys. 4.1.** Panel logowania

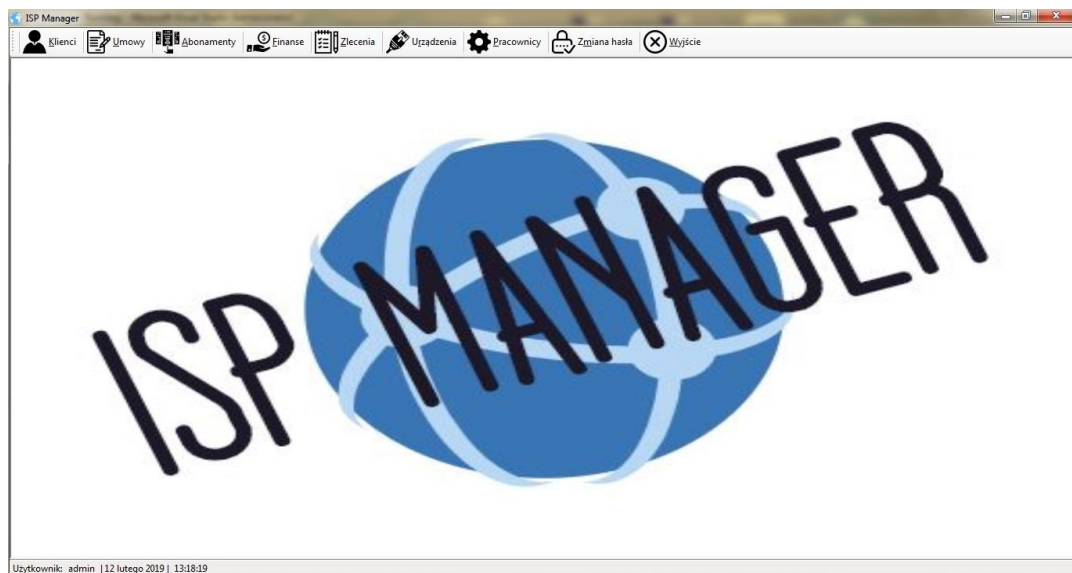
System logowania sprawdza dane wprowadzone przez użytkownika. W pierwszej kolejności nawiązywane jest połączenie z bazą danych i sprawdzenie czy podany login znajduje się w bazie. W przypadku braku połączenia lub gdy podanego loginu nie ma w bazie, zostaje wyświetlony odpowiedni komunikat. Jeżeli podany login istnieje w bazie danych to następuje weryfikacja hasła. Weryfikacja wprowadzonych danych odbywa się przy użyciu następującego kodu:

```
SqlCommand command = new SqlCommand("SELECT Password,
Klienci, Umowy, Abonamenty, Wplaty, Zlecenia,
Urzadzenia, Pracownicy FROM Users WHERE
Login=@Login", connection);
if ((err2 == null) && (pass != ""))
{
    if (Hash.Get_Hash(TPass.Text) == pass)
    {
        String sql = "UPDATE Users SET Logowanie =
@Logowanie WHERE Login = @Login";
        updateDatabase(sql);
        Glowne Glowne = new Glowne();
        Application.ExitThread();
    }
}
```

Dla bezpieczeństwa hasła w bazie danych, są hashowane kryptograficznie bez możliwości odkodowania (SHA512), więc w momencie logowania hasło podane przez użytkownika jest hashowane za pomocą procedury *Get\_Hash* z klasy *Hash* i dopiero wtedy porównywane do wartości znajdującej się w bazie [4]. Poniżej przedstawiono kod zawarty w procedurze *Get\_Hash*, służącej do hashowania dowolnego ciągu znaków:

```
public static string Get_Hash(string password)
{
    var bytes = new
    UTF8Encoding().GetBytes(password);
    byte[] hashBytes;
    using (var algorithm = new
    System.Security.Cryptography.SHA512Managed())
    {
        hashBytes = algorithm.ComputeHash(bytes);
    }
    return Convert.ToBase64String(hashBytes);
}
```

Po poprawnej weryfikacji, do bazy zostaje zapisana data i godzina logowania oraz pobrane zostają dane, do jakich komponentów ma dostęp logujący się użytkownik. Następnie zostaje wyświetlone okno główne aplikacji (Rys. 4.2)



**Rys. 4.2.** Okno główne aplikacji

Okno główne aplikacji składa się z trzech części. Większość okna zajmuje przestrzeń robocza, na której po uruchomieniu widnieje logo aplikacji. W części tej wyświetlane są poszczególne panele uruchamiane przez użytkownika. U góry

znajduje się menu (*ToolStrip* [4]) z dostępem do poszczególnych funkcji programu. W zależności od posiadanych uprawnień, użytkownik może mieć dostęp do wszystkich bądź też pojedynczych funkcji (Rys.4.3).



**Rys. 4.3.** Menu główne aplikacji

Menu składa się z następujących opcji:

- Klienci – przeglądanie, dodawanie, edycja i usuwanie klientów,
- Umowy – przeglądanie, dodawanie, edycja i usuwanie umów zawartych z klientami,
- Abonamenty – przeglądanie, dodawanie, edycja i usuwanie abonamentów dostępnych w ofercie dostawcy usług internetowych,
- Finanse – przeglądanie, dodawanie, edycja i usuwanie płatności klientów, naliczanie opłat abonamentowych, sprawdzanie salda klientów,
- Zlecenia – przeglądanie, dodawanie, edycja i usuwanie zgłoszeń serwisowych,
- Urządzenia – przeglądanie, dodawanie, edycja i usuwanie urządzeń sieciowych operatora oraz klientów, sprawdzanie dostępności urządzeń,
- Pracownicy – przeglądanie, dodawanie, edycja i usuwanie pracowników, którzy jednocześnie są użytkownikami aplikacji, nadawanie uprawnień,
- Zmiana hasła – możliwość zmiany hasła logowania,
- Wyjście – zamknięcie aplikacji.

Dodatkowo u dołu ekranu znajduje się pasek stanu (*StatusStrip* [4]), na którym wyświetlany jest login zalogowanego użytkownika oraz bieżąca data i godzina (rys. 4.4).

Użytkownik: admin | 12 lutego 2019 | 13:25:11

**Rys 4.4.** Pasek stanu

Po uruchomieniu aplikacji i poprawnym logowaniu, użytkownik może przystąpić do pracy. Dostępnych jest dziewięć paneli, w których umieszczono dostęp do poszczególnych funkcji programu. Każdy panel składa się z menu, obszaru roboczego, obszaru przeglądania danych oraz wyszukiwania. W dalszej części zostaną opisane poszczególne funkcje paneli i ich możliwości.

## 4.2. Menu paneli, przeglądanie i wyszukiwanie danych

Każdy z paneli dostępnych dla użytkownika posiada menu (*ToolStrip* [4]), w którym dostępne są podstawowe polecenia, które można wykonywać na danych dostępnych w bazie danych. Są to dodanie nowego rekordu, modyfikacja lub usunięcie istniejącego rekordu, wyczyszczenie formularza oraz zamknięcie panelu. Wygląd menu panelu został zaprezentowany na Rys.4.5.



Rys. 4.5. Menu paneli

Po uruchomieniu panelu, aktywne są opcje Dodaj, Wyczyść, oraz Zamknij. Po wypełnieniu formularza i wybraniu opcji Dodaj, następuje sprawdzenie poprawności podanych danych. Jeżeli wszystko jest w porządku, nowy rekord zostaje dodany do bazy oraz zostaje wyświetlony komunikat, informujący o udanym wykonaniu operacji. Jeżeli któraś z wprowadzonych wartości jest niepoprawna, w formularzu zostaje wyświetlony komunikat, którą wartość należy poprawić. Opcja Wyczyść powoduje wymazanie wszystkich danych wprowadzonych w formularzu, natomiast Zamknij zamyka otwarty panel. Opcje Aktualizuj oraz Usuń zostają aktywowane, gdy z tabeli w możemy przeglądać istniejące rekordy, zostanie wybrany jakiś wiersz (Rys.4.6). W momencie wybrania wiersza, formularz zostaje wypełniony informacjami z bazy. Przycisk Dodaj zostaje dezaktywowany i jest niedostępny. Użytkownik może w tym momencie zaktualizować dane w formularzu lub usunąć wybrany rekord. W momencie aktualizacji następuje sprawdzenie poprawności danych w formularzu. Jeżeli wszystko jest w porządku, dane w bazie są aktualizowane i wyświetlona zostaje informacja o udanym wykonaniu operacji. W przypadku błędu w formularzu zostaje wyświetlony komunikat przy niepoprawnej wartości. Jeżeli użytkownik wybierze opcję Usuń, zostanie wyświetlone zapytanie czy usunąć wybrany rekord. Po potwierdzeniu następuje usunięcie danych z bazy oraz wyświetlenie komunikatu o udanym wykonaniu operacji.

W chwili uruchomienia jakiegokolwiek panelu z bazy danych pobierane są informacje z bazy danych. Dane te są wczytywane do kontrolki DataGridView [5] (Rys.4.6) i wyświetlane na ekranie. Kontrolka ta zapewnia szybki i łatwy



w dostosowaniu sposób wyświetlania danych w tabelach. Do uzupełnienia tej kontroli wykorzystano następujący kod:

```
private void updateDataGrid()
{
    SqlCommand command = new SqlCommand("SELECT
    Finances.Id_Finances, Finances.Nazwa,
    Finances.Kwota, Finances.Typ,
    Finances.Id_Clients, Clients.Imie,
    Clients.Nazwisko, Finances.Data FROM Clients
    INNER JOIN Finances ON Clients.Id_Clients =
    Finances.Id_Clients", setConnection.con);
    SqlDataReader dr = command.ExecuteReader();
    DataTable dt = new DataTable();
    dt.Load(dr);
    dataGridView1.DataSource = dt.DefaultView;
    row_count = dataGridView1.RowCount;
    dr.Close();
}
```

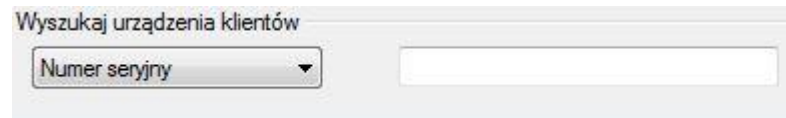
Na początku zadeklarowane jest nowe zapytanie SQL, za pomocą funkcji *SqlCommand*, wbudowanej w bibliotekę C# *System.Data.SqlClient* [9]. Zapytanie te zmienia się w zależności, który panel jest uruchomiony i jakie dane są akurat potrzebne. Następnie za pomocą funkcji *SqlDataReader* [9][10], która umożliwia odczytanie strumienia wierszy z bazy danych, zostaje wykonane podane zapytanie. Otrzymane w ten sposób rekordy zapisywane są w tablicy *DataTable* [10], a następnie wyświetlane w kontrolce *DataGridView*.

	Id	Nazwa	Typ	Adres IP
▶	1	Router 1	Router	81.15.179.1
	2	Router 3	Router	81.15.179.3
	3	Router 6	Router	81.15.179.6
	4	Router 7	Router	81.15.179.7
	5	Nadajnik Mickiewicza	Nadajnik 5GHz	172.26.4.200
	6	Nadajnik Młodzieżowa	Nadajnik 5GHz	172.24.1.200
	7	Switch Okopowa 3	Switch	172.25.3.250
	8	Switch Czerniewskiego 5	Switch	172.23.5.250

**Rys. 4.6.** Kontrolka *DataGridView*

Dane wyświetlane w kontrolce *DataGridView*, można w prosty sposób przeglądać oraz sortować według dowolnej kolumny. Jednak przy większej ilości danych, wyszukanie ręczne interesującej nas pozycji może być czasochłonne. Aby przyspieszyć tą czynność w aplikacji zaimplementowano kod, który służy do wyszukiwania interesującej nas pozycji w tabeli za pomocą danych z konkretnej

kolumny (Rys. 4.7). Opcja wyszukiwania składa się z dwóch kontroltek. Po lewej stronie znajduje się *ComboBox* [4], czyli pole wyboru, które służy do wybrania z listy nazwę kolumny, za pomocą której będziemy wyszukiwać dane. Natomiast po prawej umieszczony jest *TextBox* [4], czyli pole tekstowe, w którym użytkownik wpisuje interesującą go frazę.

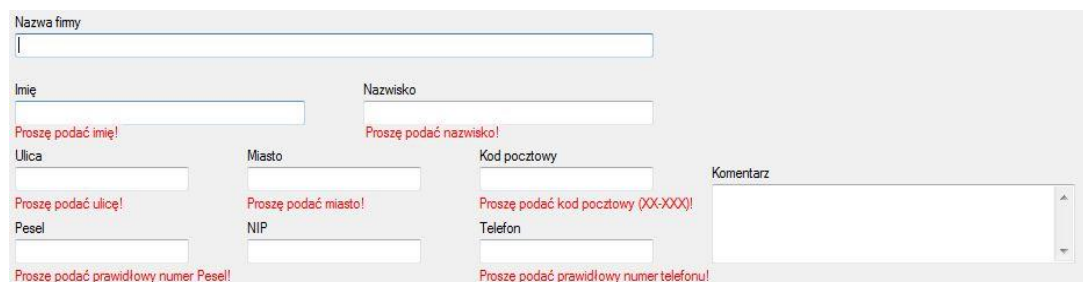


**Rys.4.7.** Pole wyszukiwania

Po wybraniu z pola wyboru nazwy kolumny, użytkownik w polu tekstowym wpisuje poszukiwane dane. W kontrolce *TextBox*, ustawione jest zdarzenie *TextChanged* [5], które uruchamia zaimplementowany kod wyszukiwania za każdym razem, gdy w kontrolce nastąpi zmiana zawartości. Tekst wpisany w *TextBox*, jest porównywany z danymi znajdującymi się w bazie danych. Jeżeli wartości zgadzają się to zostają wyświetlone tylko pasujące rekordy.

### 4.3. Panel Klienci

W panelu Klienci, użytkownik ma dostęp do danych teleadresowych klientów dostawcy usług internetowych. W tej części programu użytkownik pracuje na tabeli *Clients*, oraz na danych tam umieszczonych. Przed dodaniem nowego klienta, musi zostać wypełniony formularz, który zawiera niezbędne informacje do jego identyfikacji (Rys 4.8).



**Rys. 4.8.** Formularz panelu Klienci

W formularzu tym należy podać prawidłowe dane wymagane przez bazę danych. Pola, które obowiązkowo muszą zostać wypełnione to: Imię, Nazwisko, Ulica, Miasto, Kod pocztowy, Pesel, Telefon. Pozostałe pola są opcjonalne. W celu weryfikacji danych poszczególne pola zostały zaprogramowane do sprawdzania

zawartości. Pola Pesel i Telefon mają możliwość wpisywania tylko cyfr, oraz określona jest maksymalna ilość wpisanych znaków. Do weryfikacji Kodu pocztowego użyto funkcji *Regex* z biblioteki C# *System.Text.RegularExpressions* [6]. Utworzono wyrażenie regularne, czyli wzorzec, który opisuje łańcuch symboli:

```
Regex zip = new Regex(@"^[0-9]{2}\-[0-9]{3}$");
```

Utworzone wyrażenie regularne wymaga, aby dwa pierwsze znaki były cyframi od 0 do 9, trzeci znak to myślnik, natomiast kolejne trzy znaki to cyfry od 0 do 9. Fraza wpisana w pole Kod pocztowy jest sprawdzana pod kątem zachowania tych warunków i po ich spełnieniu jest akceptowana.

W chwili dodawania nowego klienta, następuje sprawdzenie czy klient o podanym numerze Pesel bądź NIP istnieje już w bazie. Przy modyfikacji istniejącego klienta, sprawdzane jest czy Pesel bądź NIP został zmieniony, jeżeli nastąpiła zmiana to także jest weryfikowane czy dany numer istnieje już w bazie danych. Usunięcie klienta z bazy, powoduje kaskadowe usunięcie wszystkich powiązanych z danym klientem informacji z pozostałych tabel.

#### 4.4. Panel Umowy

W panelu Umowy zawarte są informacje o umowach zawartych z klientami. W tej części aplikacji wykorzystywana jest głównie tabela *Contracts*, ale są też pobierane dane z tabel *Clients* i *Subscriptions*. Praca na danych znajdujących się tej tabeli, wykonywana jest za pośrednictwem formularz przedstawionego na Rys. 4.9.

**Rys. 4.9.** Formularz panelu Umowy

Aby dodać lub zmodyfikować umowę należy poprawnie uzupełnić formularz. Dane, które obowiązkowo należy podać to: Numer umowy, Czas Trwania, Klenci oraz Abonamenty. Umowa może mieć status obowiązująca bądź nie. W celu

wyboru statusu użyto kontrolki *CheckBox* [6], która przyjmuje wartość *True*, gdy jest zaznaczona, bądź *False*, gdy jest odznaczona. Do wyboru daty użyta została kontrolka *DateTimePicker* [9], w której wyświetlana jest bieżąca data. Wybór klienta, dla którego wprowadzamy umowę oraz abonamentu odbywa się za pomocą kontrolki *ComboBox* [4], do których wczytywana jest lista klientów z tabeli *Clients* oraz lista abonamentów z tabeli *Subscriptions*. Za wczytanie tych danych odpowiada funkcja *fillcombo* przedstawiona poniżej:

```
private void fillcombo()
{
    SqlCommand command = new SqlCommand("SELECT
    Subscriptions.* FROM Subscriptions",
    setConnection.con);
    SqlDataReader dr = command.ExecuteReader();
    while (dr.Read())
    {
        string sName = dr.GetInt32(0) + " | " +
        dr.GetString(1) + " | " + dr.GetDecimal(4);
        comboBox2.Items.Add(sName);
    }
    dr.Close();
}
```

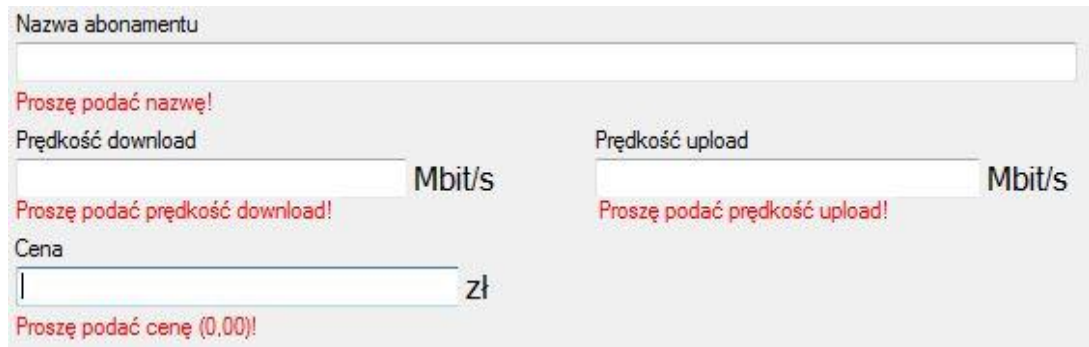
W funkcji tej wykonywane jest zapytanie SQL do bazy danych. Następnie w pętli *While* odczytana zostaje zawartość zmiennej *SqlDataReader* [9][10] i zapisana do listy *ComboBox*. Dla każdej kontrolki *ComboBox*, która pobiera dane z bazy, utworzone jest oddzielne zapytanie SQL.

W bazie danych nie mogą być przechowywane umowy o takim samym numerze, więc przy dodawaniu nowej umowy następuje sprawdzenie czy umowa o takim numerze istnieje. Natomiast przy modyfikacji najpierw sprawdzane jest czy numer umowy został zmieniony a następnie weryfikacja czy podany numer istnieje w bazie. Podczas usuwania umowy kasowane są tylko dane tej konkretnej umowy. Żadne inne tabele nie są modyfikowane.

## 4.5. Panel Abonamenty

W panelu Abonamenty użytkownik ma wgląd do danych o dostępnych abonamentach oferowanych przez dostawcę usług internetowych. Dane te przechowywane są w tabeli *Subscriptions* i w tej części aplikacji tylko na tej tabeli

wykonywane są operacje. Tak jak w pozostałych panelach do modyfikacji rekordów przeznaczony jest formularz (Rys. 4.10).



**Rys. 4.10.** Formularz panelu Abonamenty

Aby poprawnie dodać lub zmodyfikować abonament, wszystkie pola w formularzu muszą zostać prawidłowo wypełnione. Pola tekstowe Prędkość download i Prędkość upload zostały skonfigurowane w ten sposób by można było wpisywać tylko cyfry. Natomiast pole Cena jest sprawdzane za pomocą wyrażenia regularnego *Regex* [6].

Przed dodaniem lub modyfikacją abonamentu następuje sprawdzenie czy w bazie istnieje już abonament o podanej nazwie. Tabela *Subscriptions* powiązana jest relacją z tabelą *Contracts*, więc przy próbie usunięcia abonamentu z bazy sprawdzane są wszystkie umowy czy występuje na nich usuwany abonament. Jeżeli dany abonament jest przypisany do jakiejś umowy, wtedy operacja zostanie przerwana i wyświetlony zostanie odpowiedni komunikat. Kod odpowiadający za usuwanie rekordów przedstawiono poniżej:

```
private void Usun_Click(object sender, EventArgs e)
{
    DialogResult dialogResult = MessageBox.Show
        ("Usunąć abonament: " + textBox2.Text + ", ID " +
        dataGridView1.CurrentRow.Cells[0].Value.ToString(
        )
        + "?", "Komunikat", MessageBoxButtons.YesNo,
        MessageBoxIcon.Question);
    if (dialogResult == DialogResult.Yes)
    {
        String sql = "DELETE FROM Subscriptions " +
        "WHERE Id_Subscriptions = " +
        @Id_Subscriptions";
        updateDatabase(sql, 2);
        resetAll();
    }
}
```

W momencie wyboru opcji Usun, użytkownikowi ukazuje się prośba o potwierdzenie usunięcia rekordu. Po zaakceptowaniu, do zmiennej *sql* przypisane zostaje polecenie *Delete* z języka SQL [7]. W kolejnym kroku wywoływana jest klasa *updateDatabase* [5][6], w której następuje przypisanie wartości parametrom, w tym przypadku *@Id\_Subscriptions* oraz wykonane zostaje polecenie SQL. Kod ten jest wykorzystywany we wszystkich panelach, zmieniane jest tylko polecenie SQL oraz wartości parametrów w zależności, jakiej tabeli dotyczy polecenie.

#### 4.6. Panel Finanse

W panelu Finanse możemy wydzielić trzy części. Pierwsza służy do wprowadzania wpłat dokonywanych przez klientów lub naliczania pojedynczych opłat. Składa się ona z formularza, po wypełnieniu którego, do tabeli *Finances* zapisany zostaje nowy rekord. Wygląd formularza prezentuje Rys.4.11.

**Rys 4.11.** Formularz panelu Finanse

Aby pozycja została dodana do bazy wszystkie pola w formularzu muszą być prawidłowo wypełnione. Do rozróżnienia rodzaju operacji wykorzystano dwa pola typu *CheckBox* [6]. W zależności, które pole zostanie zaznaczone, do bazy zostanie wprowadzona odpowiednia wartość. Aplikacja uniemożliwia zaznaczenia obu pól jednocześnie.

Druga część panelu Finanse wykorzystywana jest do sprawdzania szczegółowego salda wybranego klienta (Rys.4.12). W części tej z kontrolki *ComboBox* [4] wybieramy interesującego nas klienta oraz rok, za jaki chcemy uzyskać informacje. Po zatwierdzeniu wyświetlone zostają nam wpłaty i naliczenia, jakie zostały zaksięgowane u danego klienta w wybranym roku. Dodatkowo po kliknięciu w przycisk z nazwą miesiąca, pojawia się okno, w którym w kontrolce

*DataGridView* [5] wyświetlone są wszystkie operacje finansowe klienta za ten okres (Rys 4.13).

Saldo klienta

Klenci: 3 | Kamil | Zed Rok: 2019 Saldo

	Naliczenia	Wpłaty	Saldo
Styczeń	25,00	0	-25,00
Luty	25,00	0	-25,00
Marzec	0	0	0
Kwiecień	0	0	0
Maj	0	0	0
Czerwiec	0	0	0
Lipiec	0	0	0
Sierpień	0	0	0
Wrzesień	0	0	0
Pazdziernik	0	0	0
Listopad	0	0	0
Grudzień	0	0	0
Podsumowanie	50,00	0	-50,00

**Rys.4.12.** Saldo klienta

Saldo

	Id	Nazwa	Kwota	Typ	Data	Imie	Nazwisko	Data
	12	Abonament Styczeń 2019r.	25,00	Naliczenie	3	Kamil	Zed	2019-01-01
	19	Abonament Luty 2019r.	25,00	Naliczenie	3	Kamil	Zed	2019-02-01

Zamknij

**Rys.4.13.** Spis operacji finansowych

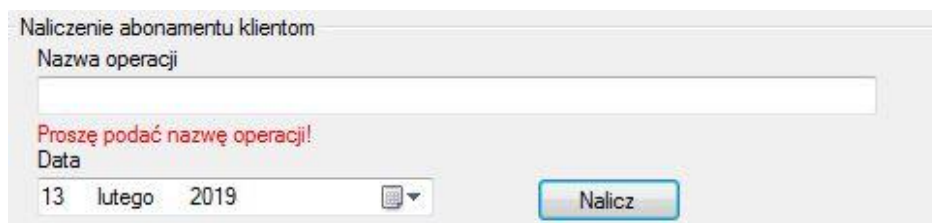


Ostatnią z możliwych operacji do wykonania w panelu Finanse to naliczenie opłat abonamentowych dla wszystkich aktywnych umów. W części tej mamy do dyspozycji formularz (Rys. 4.14), w którym wprowadzamy nazwę naliczenia oraz datę księgowania. Po wprowadzeniu tych danych i uruchomieniu naliczania aplikacja wyszukuje w bazie wszystkie umowy, które mają status „Obowiązująca”, przy użyciu następującego zapytania SQL [8]:

```
SELECT Contracts.Id_Clients, Subscriptions.Cena FROM
Contracts INNER JOIN Subscriptions ON
Contracts.Id_Subscriptions =
Subscriptions.Id_Subscriptions
WHERE (Contracts.Obowiazujaca = @Obowiazujaca)
```

W wyniku tego zapytania z tabeli *Contracts* pobrane zostają numery id klientów posiadających obowiązujące umowy oraz z tabeli *Subscriptions* pobrane zostają ceny abonamentów przypisanych do tych umów. Dane te zostają zapisane do tablic jednowymiarowych. Następnie, wykorzystując pętlę *For* [10], do tabeli *Finances* wprowadzane są opłaty abonamentowe dla każdego z klienta, który został zapisany w tablicy:

```
for (int i = 0; i < a; i++)
{
    command = new SqlCommand("INSERT INTO Finances
(Nazwa, Kwota, Typ, Data, Id_Clients) " +
"VALUES (@Nazwa, @Kwota, @Typ, @Data,
@Id_Clients)", setConnection.con);
command.Parameters.Clear();
command.Parameters.Add("@Kwota",
System.Data.SqlDbType.Decimal).Value = kwota2[i];
command.Parameters.Add("@Typ",
command.Parameters.Add("@Data",
System.Data.SqlDbType.Date).Value =
dateTimePicker2.Text;
command.Parameters.Add("@Id_Clients",
System.Data.SqlDbType.Int).Value = id_klient2[i];
int result = command.ExecuteNonQuery();
}
```

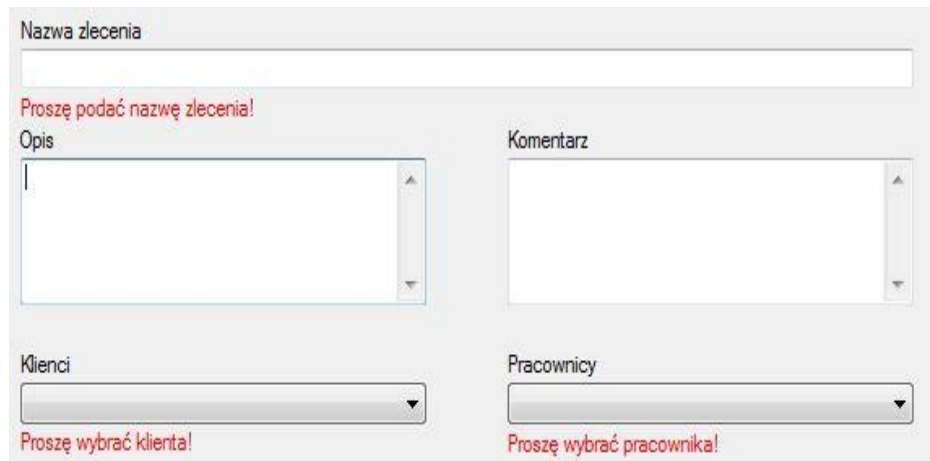
The image shows a screenshot of a Windows application window titled "Naliczenie abonamentu klientom". The window has a light gray background. At the top, there is a label "Nazwa operacji" above a text input field. Below the input field, there is a red error message that says "Proszę podać nazwę operacji!". Underneath the error message is a label "Data" above a date picker control. The date picker shows the date "13 lutego 2019". To the right of the date picker is a blue button with the text "Nalicz".

**Rys 4.14.** Formularz naliczenia opłat abonamentowych



## 4.7. Panel Zlecenia

W panelu zlecenia użytkownik pracuje na danych zawartych w tabeli *Orders*. Panel ten służy do wprowadzania, przeglądania i modyfikacji zleceń serwisowych zgłaszanych przez klientów, a także prowadzony jest harmonogram wykonywanych zleceń. Do wypełniania zgłoszenia wykorzystywany jest formularz przedstawiony na Rys.4.15.



**Rys.4.15.** Formularz panelu Zlecenia

Aby dodać lub zmodyfikować zlecenie, należy podać jego nazwę oraz wybrać z kontrolki *ComboBox* [4] klienta, który dokonał zgłoszenia i pracownika, który będzie je wykonywał. Dla zleceń już utworzonych, wybranych z pola *DataGridView* [5] do kontekstu, obok pola Nazwa zlecenia, pojawia się kontrolka *CheckBox* [6], po zaznaczeniu której, użytkownik ma możliwość zamknięcia zlecenia. Modyfikacja zleceń wykonywana za pomocą polecenia *Update* z języka SQL [7], zostało to zaprezentowane w poniższym kodzie:

```
private void Aktualizuj_Click(object sender,
EventArgs e)
{
    check_box();
    if (wszystko_ok)
    {
        String sql = "UPDATE Orders SET Nazwa =
@Nazwa, Opis = @Opis, Komentarz =
@Komentarz, Data_aktualizacji =
@Data_aktualizacji, " +
        "Id_Clients = @Id_Clients, Id_Users =
@Id_Users, Status = @Status " +
        "WHERE Id_Orders = @Id_Orders";
        id_klient = comboBox3.Text.Split('|');
```

```

        id_pracownik = comboBox2.Text.Split('|');
        this.updateDatabase(sql, 1);
        resetAll();
    }
}

```

W momencie wyboru opcji Modyfikuj, funkcja *check\_boxy* sprawdza czy dane wpisane w formularzu są prawidłowe. Po poprawnej weryfikacji, do zmiennej *sql* przypisane zostaje polecenie SQL. W kolejnym kroku wywoływana jest kasa *updateDatabase* [5][6], w której następuje przypisanie wartości parametrom, w tym przypadku są to *@Nazwa*, *@Opis*, *@Komentarz*, *@Data\_aktualizacji*, *@Id\_Clients*, *@Id\_Users*, *@Status* oraz wykonane zostaje polecenie SQL [8]. Kod ten jest niemal identyczny we wszystkich panelach, zmieniane jest tylko polecenie SQL oraz wartości parametrów w zależności, jakiej tabeli dotyczy polecenie.

#### 4.8. Panel Urządzenia

Panel Urządzenia podzielony jest na dwie części. Po lewej stronie mamy część dotyczącą urządzeń klienckich, w której operujemy na danych zawartych w tabeli *Devices\_C*. Po prawej stronie panelu umieszczona została część przeznaczona jest dla urządzeń operatora i wszystkie operacje tu wykonywane dotyczą danych z tabeli *Devices\_O*. Rys. 4.16 przedstawia formularz wykorzystywany do pracy z urządzeniami klienckimi.

**Rys.4.16.** Formularz panelu Urządzenia – urządzenia klientów

Wybór formularza, na którym będziemy pracować, odbywa się poprzez zaznaczenie pola *CheckBox* [6], umieszczonego przy nazwie formularza.

Aplikacja uniemożliwia wybrania dwóch formularzy na raz. Aby prawidłowo dodać lub zmodyfikować urządzenie klienckie należy poprawnie wypełnić wszystkie pola w formularzu oraz z kontrolki *ComboBox*, wybrać klienta, do którego jest przypisane urządzenie oraz urządzenie operatora, z którym powiązane jest dane urządzenie. Dodatkową funkcją, jaką oferuje formularz jest możliwość sprawdzenia czy dane urządzenie jest aktywne w sieci. Wykorzystane tutaj zostało polecenie *Ping*, które służy do diagnozowania połączeń sieciowych. Kod obsługujący ten proces zaimplementowany w aplikacji przedstawiono poniżej:

```
private bool PingHost(string hostname)
{
    Ping pingSender = new Ping();
    PingOptions options = new PingOptions();
    options.DontFragment = true;

    string data = "simple data";
    byte[] buffer = Encoding.ASCII.GetBytes(data);
    int timeout = 10;
    PingReply reply = pingSender.Send(hostname,
    timeout, buffer, options);

    if (reply.Status == IPStatus.Success)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Po prawidłowym wypełnieniu adresu IP, bądź też wybraniu urządzenia z listy *DataGridView* [5], klikając przycisk PING, użytkownik ma możliwość sprawdzenia czy dane urządzenie jest dostępne w sieci. W kodzie wykorzystana została funkcja *Ping* zawarte w bibliotece *System.Net.NetworkInformation*, zintegrowanej w języku C# [6]. W zależności od tego czy połączenie jest dostępne czy nie klasa *PingHost* zwraca wartość *True* lub *False*.

Formularz wykorzystywany do obsługi urządzeń operatora, jest uroszczoną wersją formularza obsługującego urządzenia klienckie. Nie ma tu pola numer seryjny oraz nie występują kontrolki *ComboBox* [4]. Pozostawiono przycisk PING,

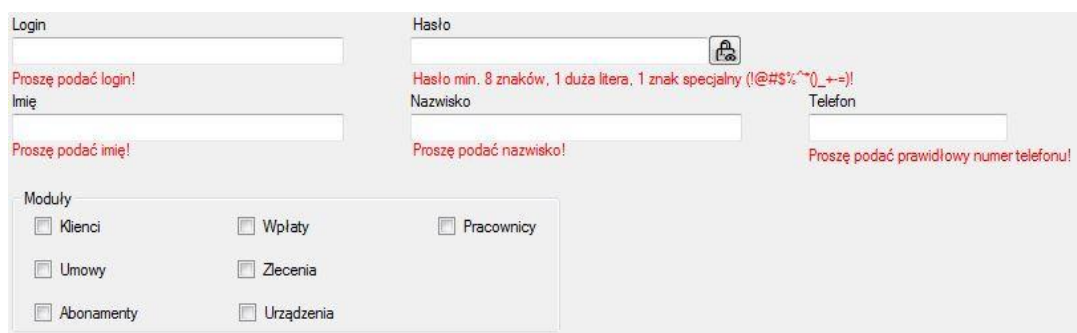
wykorzystywany do sprawdzania dostępności urządzeń w sieci. Wygląd formularza ilustruje Rys.4.17.



**Rys. 4.17.** Formularz panel Urządzenia – urządzenia operatora

## 4.9. Panel Pracownicy

Panel Pracownicy wykorzystywany jest do dodawania i modyfikacji pracowników operatora usług internetowych. Dane wprowadzone w tym panelu przechowywane są w tabeli *Users*. Przyjęto, iż każdy pracownik może być operatorem aplikacji, więc oprócz podstawowych danych identyfikujących pracownika, w formularzu (Rys 4.18) należy podać unikalny Login, hasło spełniające podane warunki (min. 8 znaków, 1 duża litera, 1 znak specjalny (!@#\$\$%^&\*( )\_+ -=), oraz wybrać do jakich paneli będzie miał dany pracownik.



**Rys. 4.18.** Formularz panel Pracownicy

Dodanie pracownika następuje po prawidłowym wypełnieniu formularza i wybraniu opcji Dodaj. Wykorzystano tutaj polecenie *Insert* z języka SQL [7]. Przed dodaniem pracownika, sprawdzane jest czy pracownik o podanym Loginie

występuje w bazie danych. Implementacja tego polecenia w kodzie aplikacji przedstawiono poniżej:

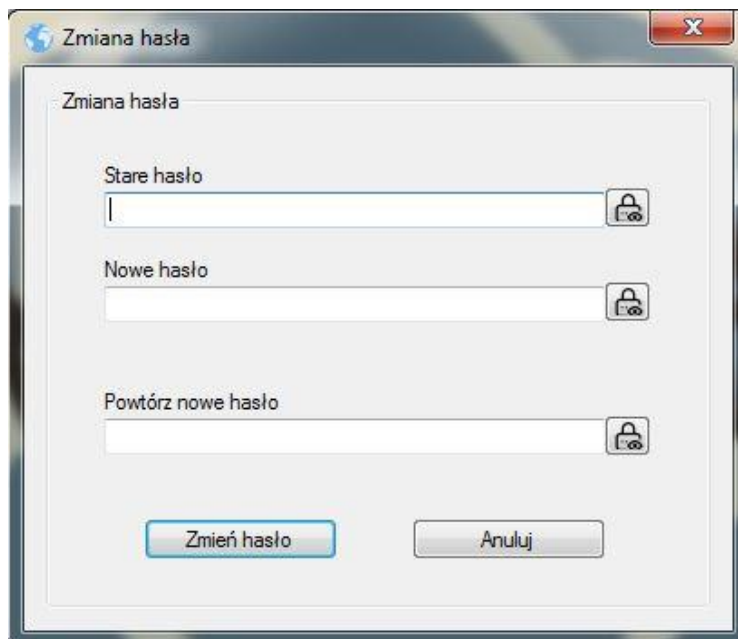
```
private void Dodaj_Click(object sender, EventArgs e)
{
    check_boxxy();
    if (wszystko_ok)
    {
        String sql = "INSERT INTO Users (Login,
        Password, Imie, Nazwisko, Telefon, Klienci,
        Umowy, Abonamenty, Wplaty, Zlecenia,
        Urzadzenia, Pracownicy) " +
        "VALUES (@Login, @Password, @Imie, @Nazwisko,
        @Telefon, @Klienci, @Umowy, @Abonamenty,
        @Wplaty, @Zlecenia, @Urzadzenia,
        @Pracownicy)";
        this.updateDatabase(sql, 0);
        resetAll();
    }
}
```

W momencie wyboru opcji Dodaj, funkcja *check\_boxxy* sprawdza czy dane wpisane w formularzu są prawidłowe. Po poprawnej weryfikacji, do zmiennej *sql* przypisane zostaje polecenie SQL. W kolejnym kroku wywoływana jest kasa *updateDatabase*, w której następuje przypisanie wartości parametrom, w tym przypadku są to *@Login*, *@Password*, *@Imie*, *@Nazwisko*, *@Telefon*, *@Klienci*, *@Umowy*, *@Abonamenty*, *@Wplaty*, *@Zlecenia*, *@Urzadzenie*, *@Pracownicy* oraz wykonane zostaje polecenie SQL [8]. Kod ten jest niemal identyczny we wszystkich panelach, zmieniane jest tylko polecenie SQL oraz wartości parametrów w zależności, jakiej tabeli dotyczy polecenie.

#### 4.10. Zmiana hasła

Ostatnią opcją dostępna dla użytkowników jest Zmiana hasła. Po wybraniu tej opcji wyświetlone jest nowe okno z formularzem (Rys. 4.19), w którym użytkownik musi podać stare hasło oraz nowe w dwóch polach. Przed zmianą, aplikacja sprawdza czy podane stare hasło jest prawidłowe, czy nowe hasło różni się od starego oraz czy spełnia podane warunki (!@#\$\$%^&\*()\_+ -=). Sprawdzane jest także czy nowe hasło wpisane w obu polach jest identyczne. Po spełnieniu wszystkich warunków hasło jest aktualizowane w bazie danych. Użytkownik informowany jest o pomyślnej zmianie hasła oraz zostaje poproszony

o ponowne uruchomienie aplikacji. Po zaakceptowaniu informacji następuje automatyczne zamknięcie aplikacji.



The image shows a standard Windows-style dialog box titled "Zmiana hasła". It contains three text input fields for passwords, each accompanied by a small lock icon indicating that the input is masked. The fields are labeled "Stare hasło", "Nowe hasło", and "Powtórz nowe hasło". At the bottom of the dialog, there are two buttons: "Zmień hasło" and "Anuluj". The "Zmień hasło" button is highlighted with a blue border, suggesting it is the default action.

**Rys. 4.19.** Zmiana hasła

## 5. Podsumowanie i wnioski

W procesie tworzenia aplikacji wykorzystano technologie cieszące się bardzo dużą popularnością wśród programistów aplikacji bazodanowych, takie jak C# i MS SQL. Można przyjąć, iż są to jedne z najbardziej niezawodnych środowisk, które doskonale ze sobą współpracują. Wykorzystanie tych technologii pozwoliło na stworzenie aplikacji bazodanowej, której zadaniem jest wspieranie pracy dostawcy usług internetowych. Aplikacja umożliwia gromadzenie informacji o klientach, umowach zawartych z klientami, abonamentach dostępnych w ofercie, rozliczeniach, zleceniach serwisowych, o urządzeniach sieciowych oraz pracownikach.

Po zakończeniu prac nad aplikacją przeprowadzono testy na podstawie których, wyciągnięto następujące wnioski:

- aplikacja dzięki prostemu i czytelnemu interfejsowi graficznemu jest łatwa i intuicyjna w obsłudze,
- dzięki optymalizacji kodu, aplikacja działa szybko i stabilnie,
- aplikacja podzielona jest na panele, które umożliwiają dalszą rozbudowę,
- wszystkie założenia funkcjonalne zostały zrealizowane i działają prawidłowo,
- stworzona baza danych, dzięki zastosowaniu relacji przechowuje tylko niezbędne dane, brak nadmiarowości.

Do uruchomienia aplikacji wymagany jest komputer z zainstalowanym systemem operacyjnym Microsoft Windows XP bądź nowszym. Dodatkowo wymagane jest następujące oprogramowanie:

- Microsoft SQL Server 2008 Express,
- Microsoft .NET Framework 3.5,
- Microsoft Visual Basic Power Packs 10.0.

W trakcie tworzenia aplikacji, głównie skupiono się na prawidłowym działaniu poszczególnych komponentów, w mniejszym stopniu skupiając się na szacie graficznej. Napisany kod jest czytelny i przejrzysty. Największą trudność w trakcie programowania sprawiło tworzenie zabezpieczeń przed błędnie

wprowadzanymi danymi w formularzach oraz implementacja skomplikowanych zapytań do bazy danych. Jedną z najbardziej czasochłonnych czynności były natomiast testy jednostkowe poszczególnych komponentów oraz rozwiązań zastosowanych w aplikacji. Jednak dzięki nim mamy pewność, że wszystkie zaimplementowane funkcje i moduły działają w sposób prawidłowy i aplikacja realizuje założone w pracy wymagania.



## Literatura

1. Kaspersky Lab Polska Sp. z o.o.,  
[https://www.securelist.pl/glossary/6091,isp\\_internet\\_service\\_provider.html](https://www.securelist.pl/glossary/6091,isp_internet_service_provider.html),  
stan z dnia 06.02.2019.
2. Pyxis P. Szkut,  
<http://pyxisisp.pl/download/Pyxis4SQL%20-%20Instrukcja%20v4.05.pdf>,  
stan z dnia 07.02.2019.
3. LMS, <https://www.lms.org.pl/about.php>, stan z dnia 07.02.2019.
4. Microsoft Visual Studio 2010 Built-in Technical documentation, *Microsoft Help Viewer 1.1 – Catalog VS\_100\_EN-US, 2010*.
5. J. Matulewski, *Visual C# 2005 Express Edition Od podstaw*, Wydawnictwo Helion, Gliwice 2006.
6. J. Liberty and D. Xie, *Programming in C# 3.0, Fifth Edition*, O'Reilly Media, Sebastopol 2007.
7. D. Kurz, *Database Internet application helping in company management*, “Poznan University of Technology Academic Journals. Electrical Engineering”, Poznań 2010, tom 64, s.175-182.
8. J. Pokorska, *Kwalifikacja E-14, Część 2. Tworzenie baz danych i administrowanie bazami*, Wydawnictwo Helion, Gliwice 2014.
9. B. Sempf, Ch. Sphar, S. R. Davis, *C# 2010 All-in-One For Dummies*, Willey Publishing, Hoboken 2010.
10. Technical documentation, API and code examples | Microsoft Docs,  
<https://docs.microsoft.com/en-us/>, stan z dnia 10.02.2019.

## Spis rysunków

<b>Rys. 3.1.</b>	Struktura bazy danych aplikacji .....	11
<b>Rys. 4.1.</b>	Panel logowania .....	13
<b>Rys. 4.2.</b>	Okno główne aplikacji .....	14
<b>Rys. 4.3.</b>	Menu główne aplikacji .....	15
<b>Rys. 4.4.</b>	Pasek stanu .....	15
<b>Rys. 4.5.</b>	Menu paneli .....	16
<b>Rys. 4.6.</b>	Kontrolka <i>DataGridView</i> .....	17
<b>Rys. 4.7.</b>	Pole wyszukiwania .....	18
<b>Rys. 4.8.</b>	Formularz panelu Klienci .....	18
<b>Rys. 4.9.</b>	Formularz panelu Umowy .....	19
<b>Rys. 4.10.</b>	Formularz panelu Abonamenty .....	21
<b>Rys. 4.11.</b>	Formularz panelu Finanse .....	22
<b>Rys. 4.12.</b>	Saldo klienta .....	23
<b>Rys. 4.13.</b>	Spis operacji finansowych .....	23
<b>Rys. 4.14.</b>	Formularz naliczenia opłat abonamentowych .....	24
<b>Rys. 4.15</b>	Formularz panelu Zlecenia .....	25
<b>Rys. 4.16</b>	Formularz panelu Urządzenia – urządzenia klientów .....	26
<b>Rys. 4.17</b>	Formularz panelu Urządzenia – urządzenia operatora .....	28
<b>Rys. 4.18</b>	Formularz panelu Pracownicy .....	28
<b>Rys. 4.19</b>	Zmiana hasła .....	30

## Oświadczenie kierującego pracą

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

.....  
Data

.....  
Podpis kierującego pracą

## OŚWIADCZENIE STUDENTA

1. Ja, niżej podpisany

Karol Budzyński

2. Numer albumu: 8096

3. Student Wydziału Informatyki i Nauk o Żywności

4. Kierunku studiów Informatyka I stopień

Oświadczam, że znam zasady przygotowywania i składania prac dyplomowych oraz zasady weryfikacji pracy dyplomowej w systemie antyplagiatowym obowiązujące w PWSiIP oraz udzielam nieodpłatnie Państwowej Wyższej Szkole Informatyki i Przedsiębiorczości w Łomży prawa do wprowadzania i przetwarzania w systemie antyplagiatowym pracy dyplomowej mojego autorstwa

pt.: ISP Manager – desktopowa aplikacja bazodanowa wspierająca pracę dostawcy usług internetowych.

Praca dyplomowa pisana pod kierunkiem promotora dr inż. Eugenii Busłowskiej.

Jednocześnie świadomy/a odpowiedzialności prawnej oświadczam, że ww. praca dyplomowa:

1. nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz. U. Nr 24, poz. 83 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
2. nie zawiera danych i informacji, które uzyskałem w sposób niedozwolony,
3. nie była podstawą nadania tytułu zawodowego ani mojej, ani innej osobie.

Świadomy/a odpowiedzialności prawnej oświadczam także, że:

1. treść pracy dyplomowej, w wersji elektronicznej zatwierdzonej przez promotora w APD (wersja do druku), jest zgodna z treścią zawartą w wydrukowanej wersji pracy, przedstawionej w procedurze dyplomowania,
2. zezwalam na nieodpłatne i bezterminowe korzystanie z przedmiotowej pracy dyplomowej w zakresie udostępniania do przeglądów, wystaw i katalogów.

Łomża, dn. .... 20.... r.

(miesiąc słownie)

.....  
(czytelny podpis studenta)