

Politechnika Śląska

Wydział Automatyki, Elektroniki i Informatyki

Kierunek Informatyka



Praca dyplomowa inżynierska

**System wspomagający zarządzanie informacją o sieci
internetowej**

Prowadzący:

dr inż. Alina Momot

Autor:

Marcin Warczyk

Gliwice 2009

Spis treści

| | |
|---|-----------|
| Wstęp | 4 |
| 1. Analiza tematu | 6 |
| 1.1 Podstawowe pojęcia | 6 |
| 1.2 Określenie wymagań | 9 |
| 2. Wybór narzędzi programowych | 13 |
| 2.1. Języki programowania i zapytań | 13 |
| 2.2. Narzędzia i technologie | 14 |
| 2.3. Środowisko programistyczne | 18 |
| 3. Specyfikacja wewnętrzna | 20 |
| 3.1. Diagram przypadków użycia..... | 20 |
| 3.2. Model danych | 21 |
| 3.3. Procedury SQL | 23 |
| 3.4. Klasy..... | 24 |
| 3.5. Wybrane algorytmy | 29 |
| 3.6. Kontrola dostępu | 33 |
| 4. Specyfikacja zewnętrzna..... | 35 |
| 4.1. Wymagania sprzętowe oraz programowe | 35 |
| 4.2. Opis instalacji | 35 |
| 4.3. Opis działania systemu | 36 |
| 5. Testowanie i uruchamianie..... | 49 |
| 5.1. Opis testowania | 49 |
| 5.2. Napotkane problemy | 50 |
| 5.3. Rozwiązanie przykładowego zadania za pomocą programu..... | 53 |
| Podsumowanie | 55 |
| Literatura | 57 |
| Dodatek A. Fragment skryptu generującego bazę danych | 58 |
| Dodatek B. Spis tabel bazy danych | 59 |
| Dodatek C. Procedury SQL..... | 61 |
| Dodatek D. Fragmenty kodu źródłowego | 64 |

Wstęp

Internet to ogólnosiwiatowa sieć komputerowa logicznie połączona w jednorodną sieć adresową opartą na protokole IP. Dostarcza lub wykorzystuje usługi wyższego poziomu oparte na telekomunikacji i związanej z nią infrastrukturze. Obecnie Internet staje się najbardziej znaczącym środkiem przekazu. W Polsce w ciągu ostatnich kilku lat znacznie wzrosła liczba osób, które z niego korzystają. W roku 2004 z Internetu korzystało w tym kraju około 8 milionów konsumentów, obecnie liczba ta jest znacznie większa. Obserwuje się intensywny wzrost osób korzystających ze stałych łączy internetowych i dostępu szerokopasmowego

Istnieje wiele firm, które zajmują się dostarczaniem połączenia z Internetem do odbiorców. Jedną z nich jest firma *CzarNet*, mająca siedzibę w miejscowości Leszczyny. Podobnie jak wiele firm tego typu, posiada ona własną bazę klientów. Dzięki temu może swobodnie kontrolować, kto korzysta z jej usług i do której z jej podsieci należy. Może również sprawdzać stan należnych płatności. W wielu wypadkach przydatny stałby się jednak także system, który pozwalałby na zarządzanie obiektami sieciowymi oraz obiektami administracyjnymi również od strony geograficznej.

Celem niniejszej pracy inżynierskiej jest stworzenie systemu wspomagającego zarządzanie informacją o sieci internetowej, realizowanego na potrzeby firmy *CzarNet*. Ma to być system inwentaryzacji obiektów sieci oraz obiektów administracyjnych razem z ich parametrami. Pozwoli to na określenie geograficznego położenia elementów sieci, podstawowych parametrów elementów oraz relacji opisywanego elementu z innymi elementami. Dzięki wspomnianemu systemowi, administrator sieci mógłby się dowiedzieć o geograficznym położeniu obiektu administracyjnego (ulica, blok mieszkalny, klatka w bloku mieszkalnym) oraz zdobyć informacje o tym, które budynki należą do danej podsieci, jacy klienci z nich korzystają czy też jak wyglądają logiczne połączenia światłowodami pomiędzy sieciami.

Każdy system informatyczny to zbiór powiązanych ze sobą elementów, którego funkcją jest przetwarzanie danych przy użyciu techniki komputerowej. W przypadku systemu zarządzania informacją o sieci internetowej tworzonego na potrzeby firmy *CzarNet* dane będą przechowywane w bazie danych, która z kolei będzie umiejscowiona na serwerze Microsoft SQL Server. Ze wspomnianym serwerem będzie łączyć się aplikacja, która będzie pozwalać wykorzystywać funkcje systemu. Planuje się również implementację aplikacji pomocniczej, która będzie modyfikowała i uzupełniała dane w bazie.

Niniejsza praca dyplomowa podzielona została na pięć rozdziałów. W pierwszym z nich przedstawiono główne cele, a także założenia, które musi spełniać tworzony system. Rozdział drugi stanowi uzasadnienie wyboru narzędzi, które zostały użyte podczas tworzenia oprogramowania, między innymi języków programowania, języków zapytań, systemu operacyjnego oraz wykorzystanych technologii i programów komputerowych. Kolejny rozdział został poświęcony specyfikacji wewnętrznej, dlatego są w nim opisane szczegóły implementacji oraz poruszony został problem kontroli dostępu do systemu. Zaprezentowano także diagram przypadków użycia danych oraz model danych. Z kolei część czwarta zawiera informacje dotyczące specyfikacji zewnętrznej. Zamieszczony został opis instalacji, menu oraz różnych zakładek i przycisków dostępnych w aplikacjach. Można odnaleźć także informacje odnośnie wymagań programowych i sprzętowych systemu oraz zalecenia dla administratora. Ostatni rozdział poświęcony jest testowaniu i uruchamianiu systemu. Opisano w nim, w jaki sposób działanie systemu było testowane, jakie pojawiły się problemy oraz w jaki sposób zostały one usunięte. Zaprezentowano w nim także zadanie rozwiązane za pomocą aplikacji. Pracę zamykają podsumowanie, bibliografia, a także dodatki.

1. Analiza tematu

W tym rozdziale pracy omówione będą podstawy teoretyczne tworzonego systemu związane z sieciami komputerowymi. Przedstawiona zostanie krótka charakterystyka firmy *CzarNet*, na której zlecenie powstanie system, oraz informacje dotyczące topologii sieciowych i okablowania sieciowego, które zostaną wykorzystane podczas implementacji systemu. Rozdział zawiera również opis wymagań systemu zarówno funkcjonalnych jak i нефункциональных.

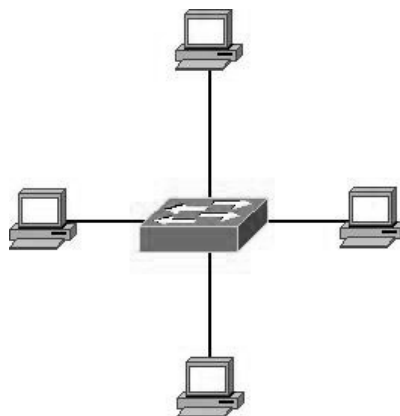
1.1 Podstawowe pojęcia

Firma *CzarNet*, która zleciła stworzenie systemu zarządzania informacją o sieci internetowej ma swoją siedzibę w miejscowości Leszczyny. Już od kilku lat zajmuje się dostarczaniem Internetu do odbiorców w swoim obszarze sieci (rozumianym jako geograficzne terytorium, na którym działa ta firma, obejmujące miejsca zamieszkania wszystkich jej klientów). W firmie wykorzystywano do tej pory jedynie bazę klientów, przy wykorzystaniu serwera MySQL, w której gromadzono ich wszystkie dane, natomiast system do zarządzania informacją o sieci internetowej nie był stosowany. Projektowany system miałby zarządzać geograficznym położeniem obiektu administracyjnego (ulica, blok mieszkalny, klatka w bloku mieszkalnym) oraz informować o tym, które budynki należą do danej podsieci, jacy klienci z nich korzystają czy też jak wyglądają logiczne połączenia światłowodami pomiędzy sieciami

Poniżej znajduje się omówienie pod względem teoretycznym kilku zagadnień, które będą wykorzystywane w tworzonym oprogramowaniu. Dotyczą one topologii sieciowych oraz rodzaju okablowania sieciowego wykorzystywanego przez firmę *CzarNet*.

Topologie sieciowe

Obecnie najczęściej stosowany sposób połączenia komputerów w sieci to topologia gwiazdy [11]. Jest ona stosowana również przez firmę *CzarNet*. Charakteryzuje się tym, że kable sieciowe połączone są w jednym wspólnym punkcie, w którym znajduje się przełącznik (switch). Komputery są przyłączone do segmentów kabla wychodzących z centralnej lokalizacji albo przełącznika. Schematyczne przedstawienie topologii gwiazdy prezentuje rysunek 1.1.



Rysunek 1.1. Połączenie komputerów w topologii gwiazdy

Rozwinięciem tej technologii jest topologia gwiazdy rozszerzonej, która oparta jest o topologię gwiazdy [12]. W tej topologii każde z urządzeń końcowych działa jako urządzenie centralne dla własnej topologii gwiazdy. Topologia ta stosowana jest głównie w przypadku rozbudowanych sieci lokalnych, gdy obszar, który ma być pokryty siecią jest większy niż pozwala na to topologia gwiazdy.

Firma *CzarNet*, która będzie korzystała ze stworzonego systemu również stosuje topologię gwiazdy rozszerzonej w połączeniach komputerów w sieci ze względu na jej największą niezawodność spośród wszystkich dostępnych topologii. System zarządzania informacją o sieci internetowej będzie uwzględniać ten sposób konfiguracji w sposobie działania.

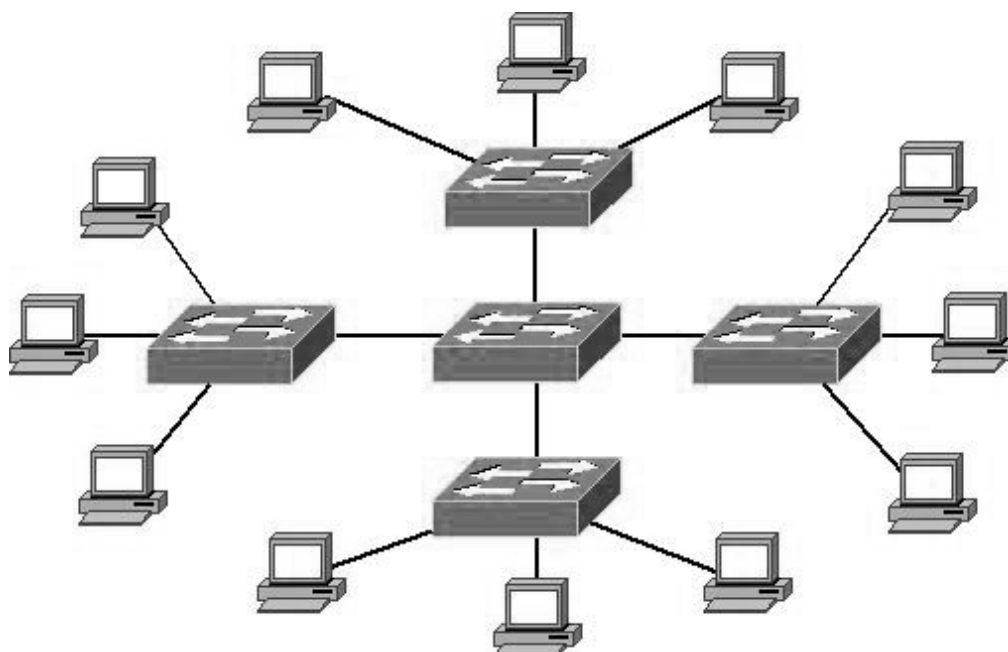
Dostawcy Internetu zazwyczaj łączą komputery w sieci stosując wyżej wymienione rodzaje topologii ze względu na liczne korzyści. Można do nich zaliczyć między innymi:

- dużą przepustowość,
- gdy przestaje działać jeden komputer, cała sieć funkcjonuje dalej,
- łatwo zlokalizować uszkodzenia, ze względu na centralne sterowanie,
- ograniczenie liczby urządzeń, które muszą być podłączone z centralnym węzłem.

W topologii gwiazdy, kabel sieciowy z każdego komputera jest podłączony do centralnego urządzenia zwanego przełącznikiem, który łączy kilka komputerów. Na przykład w bloku mieszkalnym istnieje sieć komputerowa. Sygnał źródłowy dochodzi do przełącznika za pomocą światłowodu (o określonej ilości żył) a stamtąd sygnał dociera do kilku klientów mieszkających w danym bloku. Wszyscy oni należą wtedy do tej samej sieci. Korzystają ze wspólnego przełącznika. W topologii gwiazdy sygnał jest przesyłany z komputera przez przełącznik, do wszystkich komputerów w sieci.

Awaria jednego komputera powoduje, że tylko on nie będzie mógł wysyłać i odbierać danych. Reszta sieci będzie funkcjonować bez żadnych zakłóceń. Wadą stosowania tej

topologii jest to, że gdy awarii ulegnie przełącznik, to wtedy cała sieć (wszystkie komputery, które są podpięte do danego przełącznika) przestaje funkcjonować. Schematyczne przedstawienie topologii gwiazdy rozszerzonej prezentuje rysunek 1.2.



Rysunek 1.2. Połączenie komputerów w topologii gwiazdy rozszerzonej

Inne rodzaje dostępnych topologii to: magistrala, pierścień, podwójny pierścień, topologia hierarchiczna, siatka, topologia liniowa. Każda z nich jest stosowana w różnych okolicznościach w zależności od zapotrzebowania. Mogą też istnieć technologie łączone, na przykład topologia magistrali i gwiazdy. Takie rozwiązanie jest niekiedy potrzebne, ponieważ gdy konfiguracji gwiazdy nie da się bardziej rozbudować, można dodać drugą gwiazdę i połączyć obie topologie gwiazdy w układzie magistrali.

Okablowanie sieciowe

Komputery łączą się w sieci razem za pomocą kabli sieciowych, aby przysyłać sygnały między komputerami. Okablowanie, które istnieje na rynku różni się swoimi możliwościami i jest podzielone na różne kategorie określające prędkości transmisji danych oraz odporność na zakłócenia. Trzy główne typy okablowania to:

- skrętka,
- kabel koncentryczny,
- światłowód.

Skrętka jest kablem sygnałowym, który zbudowany jest z jednej lub więcej par skręconych ze sobą przewodów miedzianych. Każda z par posiada inną długość skręcenia w

celu obniżenia zakłóceń wzajemnych. Skręcenie przewodów powoduje równocześnie zawężenie pasma transmisyjnego. Z kolei kabel koncentryczny to przewód miedziany otoczony izolacją, wspólnym ekranem oraz zewnętrzną koszulką ochronną. Zarówno skrętka jak i kabel koncentryczny są wykorzystywane jako medium transmisyjne między innymi w sieciach Ethernet.

Światłowód transmituje cyfrowe dane przez włókno optyczne w formie modulowanych impulsów świetlnych [6]. Ponieważ sygnały elektryczne nie są w tym wypadku przesyłane więc sygnał nie może być podsłuchany. Korzystanie więc z tego rodzaju okablowania jest bardzo bezpieczne. Światłowód jest przeznaczony do bardzo szybkich transmisji dużych ilości danych. Wszystko to jest możliwe dzięki temu, że sygnał jest transmitowany bardzo szybko i z bardzo małymi zakłóceniami. Wadą stosowania światłowodu jest łatwość uszkodzenia podczas nieostrożnego montażu.

Firma *CzarNet* wykorzystuje światłowody aby połączyć między sobą sieci komputerowe. W systemie wspomagającym zarządzanie informacją o sieci internetowej tworzonym dla tej firmy będzie istniała możliwość tworzenia logicznych połączeń światłowodowych pomiędzy sieciami.

1.2 Określenie wymagań

Poniżej opisano wymagania funkcjonalne oraz нефункционалне, jakie są stawiane przed projektowanym systemem. Wymagania te opisują usługi, które system ma udostępniać i ograniczenia, które musi spełniać. Są podstawą do stworzenia oprogramowania. Opisano również wymagania programowe oraz sprzętowe. Wymagania dla większej przejrzystości zostały podzielone na logiczne grupy.

Użytkownicy i ich konta

- Do wybranego obszaru sieci może zalogować się tylko użytkownik, który posiada prawa dostępu do tego obszaru.
- System umożliwia logowanie przy użyciu trzech poziomów dostępu:
 - super administrator,
 - administrator,
 - użytkownik.
- Istnieje możliwość zarządzania kontami użytkowników. Tylko administrator może realizować te funkcje.

- W systemie każdy obszar sieci zarządzany przez daną firmę musi mieć swojego właściciela.
- System umożliwia wylogowywanie użytkowników.
- System powinien umożliwiać działanie na więcej niż jednym stanowisku roboczym.

Dodawanie, usuwanie i edycja elementów

- System zapewnia możliwość dodawania nowego obszaru sieci. Tworzenie nowego obszaru sieci jest związane z tworzeniem konta super administratora, który będzie właścicielem tego obszaru.
- Usuwać obszar sieci oraz konta administratorom może tylko super administrator.
- Istnieje możliwość dodawania, edycji i usuwania obiektów administracyjnych, klientów oraz obiektów związanych z siecią komputerową.
- System umożliwia dodawanie nowych typów urządzeń w sieci wraz z konkretnymi modelami tych urządzeń.
- System umożliwia rysowania bloków i ulic (obiekty administracyjne i współrzędne tych obiektów na mapie). Dodawanie wierzchołków bloków mieszkalnych i ulic następuje dopiero po wpisaniu wszystkich niezbędnych danych tekstowych dotyczących nowo powstającego obiektu. System umożliwia również anulowanie rysowania bloków i ulic
- Usuwanie serwera skutkuje tym, że zostają usunięte również wszystkie połączenia sieciowe z danego obszaru sieci i należące do niej podsieci.
- W ramach jednego obszaru sieci może znajdować się tylko jedna serwerownia.
- Jeżeli w danym obszarze sieci nie ustalono jeszcze serwerowni, to system uniemożliwia dodawanie nowych połączeń sieciowych.
- Istnieje możliwość usuwania wszystkich obiektów związanych z siecią i należących do danego obszaru sieci.
- System umożliwia usuwanie wszystkich informacji dotyczących danego obszaru sieci (obiektów administracyjnych, sieciowych, klientów) tylko przez super administratora. Po wykonaniu takiej czynności powinno nastąpić wylogowanie administratora zarządzającego usuniętym obszarem sieci.

Wyświetlanie, drukowanie i wyszukiwanie danych

- Możliwe jest wyświetlanie położenia obiektów administracyjnych (bloki mieszkalne, ulice) oraz logicznego połączenia światłowodowego pomiędzy sieciami.

- Istnieje opcja wyboru widoku map załadowanych przez użytkownika, na których będą rysowane obiekty. Każdy obszar sieci może posiadać mapę oraz zdjęcie satelitarne.
- Istnieje możliwość wydrukowania listy wszystkich klientów należących do danego obszaru sieci lub klientów należących do danej ulicy.
- Istnieje możliwość wydrukowania listy wszystkich klientów należących do danego obszaru sieci lub klientów należących do podanej ulicy, którzy w danej chwili nie mają połączenia z Internetem.
- System umożliwia podgląd wydruków.
- System umożliwia wyszukiwanie bloków, mieszkalnych, ulic, klientów, wszystkich sieci, tylko niepołączonych sieci.
- Jeżeli zostanie znaleziony przynajmniej jeden szukany obiekt to powinny zostać wyświetlone informacje szczegółowe na jego temat, a jeśli jest to obiekt administracyjny to powinien zostać narysowany.
- Istnieje możliwość przeglądania informacji o klatkach w blokach za pomocą klawiszy klawiatury.
- Istnieje możliwość sprawdzania stanu połączenia z sieciami oraz z poszczególnymi klientami.
- Istnieje usługa w systemie, która umożliwia wykrywanie współrzędnych na mapie terenu, po tym jak użytkownik wybierze jakiś punkt na tej mapie. Jeżeli wybrany punkt mieści się w obszarze bloku mieszkalnego, to zostaną wyświetlone szczegółowe informacje na temat tego bloku.
- System umożliwia podgląd ważnych zdarzeń i informacji za pomocą paska statusu. Znajdują się na nim następujące informacje: aktualna pozycja kursora, aktualnie zalogowany użytkownik, lokalizacja serwera, nazwa ostatnio wykonanej czynności.
- Użytkownik ma możliwość zmienić w dowolnym momencie lokalizację pliku graficznego, który jest wczytywany jako mapa.
- System powinien dokonywać poprawnych wydruków na drukarkach atramentowych.

Wymagania czasowe, programowe i sprzętowe

- Funkcja wyszukiwania połączeń powinna zwracać wynik w czasie opisanym przez poniższe równanie:

$$t < 120\text{ms} * x,$$

gdzie:

t – czas trwania wyszukiwania niepołączonych (lub połączonych) z siecią klientów

x – ilość połączeń do przetestowania (na przykład ilość klientów)

- Czas testowania połączenia z serwerem nie powinien przekraczać 30 sekund.
- Do poprawnego funkcjonowania wymagany jest system operacyjny Microsoft Windows XP w wersji Professional lub Home Edition.
- System poprawnie funkcjonuje, gdy na komputerze zainstalowano .NET Framework 3.5 oraz Microsoft SQL Server 2005.
- Baza danych jest umieszczona na serwerze (serwer relacyjnych baz danych Microsoft SQL Server). Aplikacja łączy się z serwerem i ma uzyskiwać dostęp do bazy.
- Minimalnie wymagany jest procesor: 1.5GHz oraz pamięć RAM: 196MB.

2. Wybór narzędzi programowych

W niniejszym rozdziale poświęcono uwagę opisowi narzędzi programowych, które były wykorzystywane podczas tworzenia systemu dla firmy *CzarNet*.. Są to narzędzia, które umożliwiły zaprojektowanie bazy danych, stworzenie diagramów oraz implementację. Uzasadniono również wybór poszczególnych języków programowania, języków zapytań oraz środowiska programistycznego. Wszelkie czynności związane z tworzeniem systemu były wykonywane z użyciem systemu operacyjnego Microsoft Windows XP Professional.

2.1. Języki programowania i zapytań

Wiele systemów jest obecnie implementowanych za pomocą obiektowych języków programowania takich jak C#, Java oraz C++. Każdy z tych języków posiada zalety, ze względu na specyficzne właściwości. Języki zapytań są stosowane do formułowania zapytań w odniesieniu do bazy danych. Do najważniejszych należą standardy języka SQL oraz xBASE. W tym podrozdziale opisano cechy i zalety wybranych języków programowania oraz języków zapytań.

Microsoft Visual C# .NET

Język programowania C#, który jest częścią platformy .NET, został wybrany przede wszystkim ze względu na jego znajomość przez autora. Jest to jeden z obecnie najczęściej stosowanych, najpopularniejszych obiektowych języków programowania, mających wiele zalet. Jedną z nich jest konsekwentna obiektowość. Oznacza to, że wszystko, co znajduje się w pamięci jest obiektem (jest instancją jakiejś klasy lub struktury). Istotą każdego obiektowego języka programowania jest obsługa tworzenia i używania klas [2]. Klasy pozwalają definiować nowe typy, co umożliwia rozszerzenie języka w celu lepszego dopasowania go do problemu, który programista ma do rozwiązywania. Język C# zawiera kluczowe słowa, które służą do deklarowania nowych klas, a także do obsługi hermetyzacji, dziedziczenia i polimorfizmu – trzech podstawowych właściwości programowania obiektowego.

Biblioteki platformy .NET, w przeciwieństwie do często używanych funkcji *WinAPI*, zostały przygotowane w technologii obiektowej. Ułatwia to znacznie ich wykorzystanie. W języku C# nie ma typów prostych ani funkcji. Są natomiast klasy, ich pola, właściwości i metody. Pozwala to na łatwe uporządkowanie kodu programu. Oprócz tego, język C# udostępnia pełną obsługę delegatów, umożliwiających pośrednie wywołanie metod.

Kolejną zaletą jest fakt, że kontrolą pamięci zajmuje się platforma .NET. Pisząc w języku C#, programista nie musi się przejmować, czy pamięć zarezerwowana przez każdy obiekt została zwolniona i co się z nią dzieje po zamknięciu aplikacji. Te czynniki pozwalają uniknąć wielu błędów w czasie implementacji. Pisanie programów z użyciem tego języka jest bardzo wygodne. Jest tak również między innymi dzięki mechanizmowi podpowiadaniu kodu. Na przykład oprócz podpowiadanej nazwy konkretnej metody danej klasy wyświetlana jest również informacja o poszczególnych parametrach, jakie mają zostać przekazane oraz opis działania tej metody. Jeszcze jedną zaletą C# jest działanie we wspólnym środowisku uruchomieniowym, które umożliwia współpracę z innymi kompilatorami, przez co część projektu może zostać zaimplementowana w innych językach [13].

SQL i LINQ

Język SQL to najpopularniejszy język do pobierania informacji o danych z baz i dokonywania na nich zmian [2]. Ten język był wykorzystywany w implementacji systemu tworzonego w ramach niniejszej pracy na przemian z językiem LINQ. Używany był również podczas testowania poprawności pobierania danych z bazy.

LINQ (*Language INtegrated Query*) to zintegrowany język zapytań dostępny w .NET wersji 3.5. Główne cechy a zarazem zalety tego języka to [10]:

- zapytania do bazy stają się częścią języka,
- dla bardzo dużej ilości danych operacje pobierania danych z bazy są szybsze niż z wykorzystaniem języka SQL,
- zapytania nie są pisane w cudzysłowach,
- sprawdzanie poprawności następuje w czasie kompilacji,
- obecność mechanizmu podpowiadania kodu (*IntelliSense*),
- kontrola typów danych i ich konwersji dla pobieranych danych.

2.2. Narzędzia i technologie

Obecnie istnieje wiele narzędzi i technologii, które znacznie ułatwiają tworzenie systemów informatycznych. Za ich pomocą można między innymi modelować system, implementować go, wykonać niezbędne diagramy oraz tworzyć bazy danych na serwerach. Poniżej opisano platformę .NET, ADO.NET, SQL Server 2005 oraz Enterprise Architect, które będą wykorzystywane przy implementacji systemu.

Platforma .NET Framework

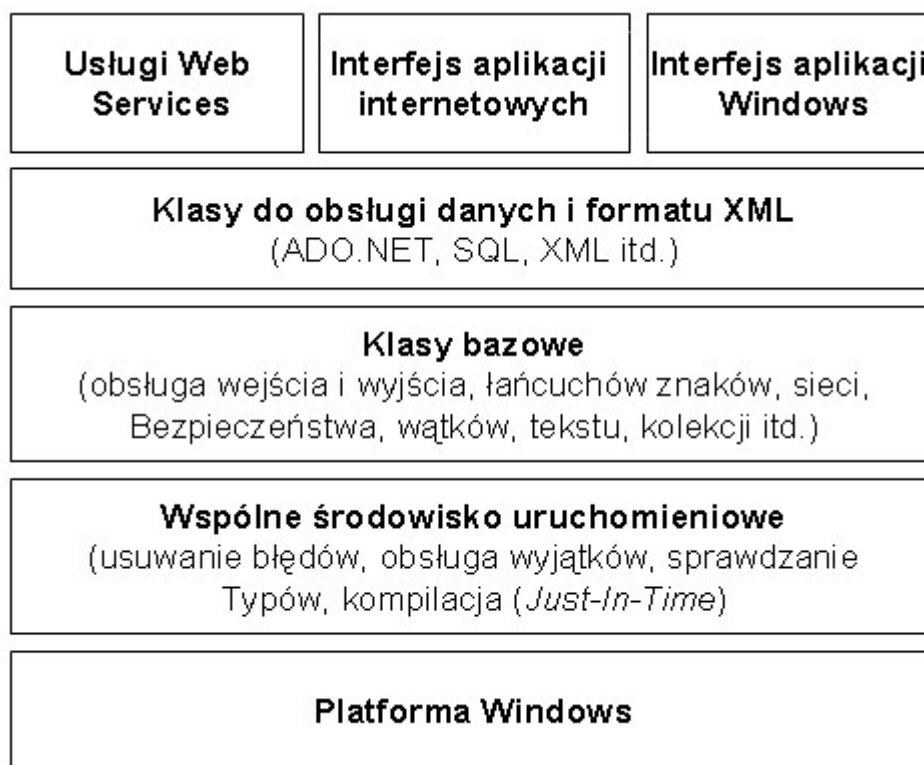
Platforma .NET to zintegrowane środowisko, które umożliwia bezproblemowe wytwarzanie i uruchamianie aplikacji. Platforma ta składa się ze wspólnego środowiska uruchomieniowego połączonego z biblioteką klas bazowych [2], która zapewnia podstawową funkcjonalność powstających aplikacji [3]. Wspólne środowisko uruchomieniowe zarządza całym cyklem życia aplikacji: lokalizuje kod, kompiluje go, wczytuje klasy, zarządza wykonaniem aplikacji oraz zapewnia zarządzanie pamięcią. Schematyczne przedstawienie architektury platformy .NET prezentuje rysunek 2.1.

Biblioteka klas bazowych dostarcza odpowiednich klas na przykład do obsługi dostępu do bazy danych różnego typu (między innymi zastosowanej w niniejszej pracy typu bazy SQL Server 2005), formularzy, operacji graficznych. Biblioteki wielokrotnego użytku definiują typy, które są udostępniane aplikacjom wykonywanym w środowisku uruchomieniowym platformy .NET.

Celem twórców platformy .NET było między innymi to, żeby zapewnić środowisko przenośne, które miałyby bazować na certyfikowanych standardach. Dzięki temu może być zarządzane przez dowolny system operacyjny. Standard infrastruktury wspólnego języka (*Common Language Infrastructure*) - CLI - definiuje niezależne od platformy środowisko do wykonywania kodu wirtualnego. CLI nie preferuje żadnego z systemów operacyjnych. Dzięki temu może mieć zastosowanie zarówno w systemie Windows jak i Linux. Jednak zastosowanie w niniejszej pracy bazy danych typu SQL Server 2005 ogranicza uruchomienie stworzonego oprogramowania tylko do systemu Windows. Jednym z najważniejszych elementów tego standardu jest definicja wspólnego języka pośredniego, który musi być generowany przez kompilatory zgodne ze standardem CLI, a także system typów (CTS – *Common Type System*), definiujący typy danych obsługiwane przez wszystkie zgodne języki programowania. CTS obsługuje ogólne pojęcia klas, interfejsów i delegatów.

Ponadto .NET zawiera wspólną specyfikację języka (*Common Language Specification*), udostępniającą zestaw podstawowych reguł niezbędnych do integracji języków programowania. Technologia ta nie jest związana z żadnym konkretnym językiem programowania, a programy mogą być pisane w jednym z wielu języków.

Podczas tworzenia systemu zarządzania informacją o sieci internetowej dla firmy *CzarNet* skorzystano z .NET Framework w najnowszej wersji czyli 3.5 ze względu na zaimplementowanie w niej LINQ co było często wykorzystywane w projekcie.



Rysunek 2.1. Architektura .NET Framework

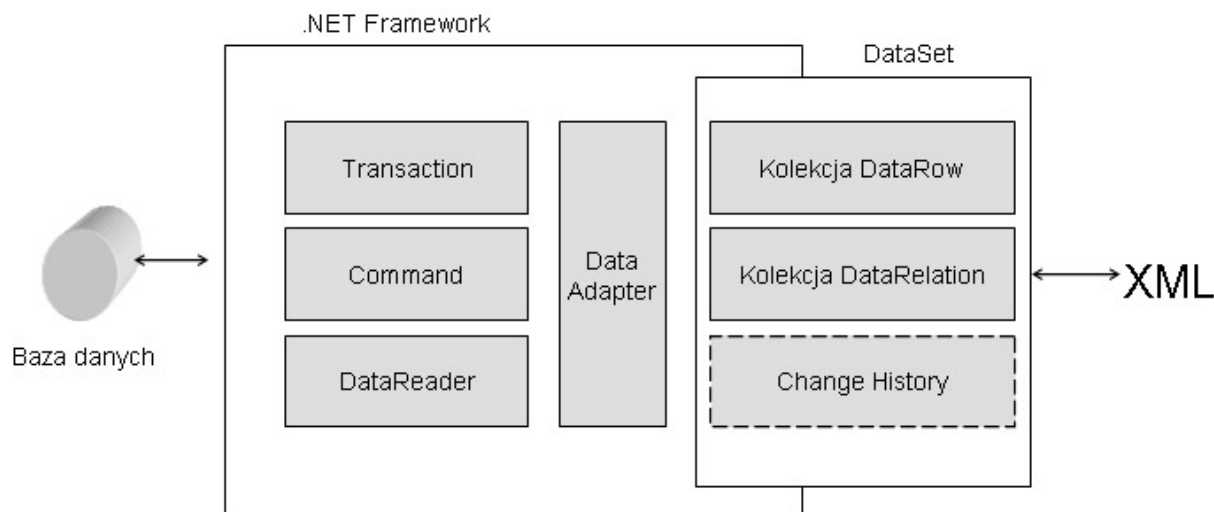
ADO.NET

Architektura ADO.NET opiera się na elastycznym zestawie klas, które umożliwiają dostęp do danych z poziomu zarządzanego środowiska .NET. Te klasy pozwalają na dostęp do różnych źródeł danych – relacyjne bazy danych, pliki XML, arkusze kalkulacyjne, pliki tekstowe. Dostęp do danych odbywa się poprzez interfejsy API. Ułatwia to wiele zadań programistycznych, takich jak tworzenie klientów baz danych.

ADO.NET udostępnia dwa modele połączenia z bazą danych – model połączeniowy i bezpołączeniowy. Model połączeniowy opiera się na dostępie do danych rekord po rekordzie, co wymaga otwarcia i utrzymywania połączenia ze źródłem danych. Dzięki zastosowaniu modelu bezpołączeniowego nie jest wymagane utrzymywanie aktywnego połączenia ze źródłem danych. ADO.NET pozwala korzystać z połączeń oszczędnie. W modelu bezpołączeniowym aplikacje utrzymują połączenie z bazą danych tylko na czas odczytywania lub zapisywania danych. Z tego wynika ważna zaleta – oszczędzane są zasoby systemowe, ponieważ obciążenie spowodowane utrzymywaniem otwartych połączeń zmniejsza ogólną wydajność aplikacji [2].

Obiekty ADO.NET łączą się z bazą, aby pobrać z niej informacje, a następnie łączą się ponownie, aby dokonać aktualizacji bazy po wprowadzeniu zmian. Udostępniają pobrany

podzbiór danych podczas ich odczytu i zapisu. Taki rodzaj połączenia jest wykorzystywany w systemie stworzonym w ramach niniejszej pracy. Łączenie ze źródłem danych jest realizowane na przykład na podstawie odpowiednich komend języka SQL. .NET Framework udostępnia klasy [5], których funkcjonalność pozwala na ładowanie danych a *DataSet* pełni rolę bazy danych w pamięci programu (udostępnia również kolekcje pokazane na rysunku 2.2). *DataAdapter* służy jako pośrednik między obiektem *DataSet* a źródłem danych.



Rysunek 2.2. Architektura ADO.NET

SQL Server 2005

Microsoft SQL Server 2005 to bardzo popularny system zarządzania bazą danych typu klient - serwer. Strona serwera zapewnia aplikacji zabezpieczenia, odporność na uszkodzenia, wydajność, współbieżność i wiarygodne kopie zapasowe. Strona klienta dostarcza interfejsu użytkownika. SQL Server jest w tym modelu stroną serwera [4]. Baza danych na serwerze SQL Server zawiera nie tylko pierwotne dane, ale także strukturę bazy danych, wszelkie indeksy, zabezpieczenia bazy i być może inne obiekty takie jak widoki lub procedury składowane związane z określoną bazą [8]. Klientem jest oprogramowanie, którego stworzenie było celem niniejszej pracy.

W SQL Server 2005 wprowadzono SQL Server Management Studio — nowy zintegrowany pakiet narzędzi do zarządzania. Wprowadzono też funkcje zabezpieczeń takie jak szyfrowanie baz danych, bezpieczne ustawienia domyślne, wymuszanie zasad haseł, szczegółowa kontrola uprawnień oraz rozszerzony model bezpieczeństwa.

Inne znane systemy zarządzania bazą danych to na przykład *Oracle* czy *MySQL*. Jednak autor nie miał wcześniej doświadczenia w ich wykorzystaniu w przeciwieństwie do narzędzia opisywanego w tym podpunkcie. Firma Microsoft udostępniła edycję darmową *SQL Server 2005 Express* do dowolnego zastosowania. Ten system został wybrany przez autora ze względu na wymienione wcześniej zalety oraz ze względu na łatwą integrację z platformą .NET. Obiekty baz danych SQL Server 2005 mogą być tworzone przy użyciu znanych języków, takich jak Microsoft Visual C# .NET.

Enterprise Architect

Narzędzie to używane jest do komputerowego wspomagania projektowania oprogramowania (oprogramowanie typu CASE - Computer-Aided Software Engineering). Za jego pomocą można modelować systemy informatyczne przy użyciu języka UML (Unified Modeling Language, czyli Ujednolicony Język Modelowania). Pozwala na gromadzenia wymagań jakie powinien mieć system informatyczny, analizowanie, modelowanie biznesowe. Za jego pomocą można tworzyć diagramy przypadków użycia, diagramy klas, komponentów, projektowania [9]. Zapewnia sporo elementów graficznych i prosty w obsłudze interfejs. Korzystając z Enterprise Architect można również w łatwy sposób wygenerować raport w postaci pliku *.html*, w którym zamieszczone będą wszystkie zaprojektowane schematy, modele, diagramy.

Podczas tworzenia systemu zarządzania informacją o sieci internetowej dla firmy *CzarNet* oprogramowanie to zostało wykorzystane do stworzenia diagramu przypadków użycia oraz modelu danych. *Enterprise Architect* to jedno z wielu narzędzi tego typu na rynku. Bardzo znane i często używane są również *Power Designer* oraz *IBM Rational Rose*. Autor wybrał jednak *Enterprise Architect* przede wszystkim ze względu na doświadczenie w jego używaniu.

2.3. Środowisko programistyczne

Podczas projektowania systemu dla firmy *CzarNet* wykorzystano środowisko programistyczne Microsoft Visual Studio .NET 2008. W skład narzędzi udostępnianych przez środowisko wchodzi między innymi wykorzystywany w pracy język programowania Microsoft Visual C#. Wykorzystano wersję 2008, ponieważ jak na razie tylko ona udostępnia .NET Framework w wersji 3.5, która była niezbędna podczas implementacji. Wcześniejsza wersja nie udostępniała niektórych ważnych bibliotek (takich jak *System.Linq*).

To środowisko zostało wybrane przez autora pracy ze względu na wiele zalet, które posiada. Najbardziej ogólną jednostką organizacyjną w Visual Studio jest *Solution* (czyli „rozwiązanie”). Pozwala to zgrupować dowolną ilość projektów w jednej, bardzo łatwej do zarządzania strukturze [1]. Jest to narzędzie, dzięki któremu można utrzymać kontrolę nad projektami. Było to bardzo przydatne podczas tworzenia oprogramowania, dlatego, że składa się ono z kilku projektów – dokładnie z trzech.

Program Visual Studio 2008 wnosi wiele nowych technologii, w tym ulepszone cechy języka i danych. Dzięki nim programista może w szybki sposób tworzyć połączone aplikacje oraz łatwiej pokonywać problemy programistyczne. Warto również w tym miejscu wspomnieć o usłudze MSDN, która udostępnia dokumentację, pomoc techniczną oraz aktualne oprogramowanie przeznaczone dla programistów tworzących programy przy użyciu produktów firmy Microsoft.

Omawiane środowisko programistyczne udostępnia mechanizm *IntelliSense*, który asystuje programiście w trakcie pisania kodu programu [1]. Odpowiada on za wyświetlanie listy składowych klasy natychmiast po zakończeniu wpisywania jej nazwy i później. Wybranie klawiszy *Ctrl+ Spacja* powoduje wywołanie *IntelliSense* w dowolnym momencie a nie dopiero po zakończeniu wpisywania nazwy klasy. Użycie tych samych klawiszy w pustym wierszu spowoduje wyświetlenie pełnej listy dostępnych składowych w bieżącym kontekście. Jest to dobry sposób na znalezienie nazwy zmiennej, której programista mógł zapomnieć. Z uwagi na wiele wymienionych wyżej zalet to właśnie Microsoft Visual Studio .NET 2008 został wybrany przez autora jako środowisko programistyczne.

3.Specyfikacja wewnętrzna

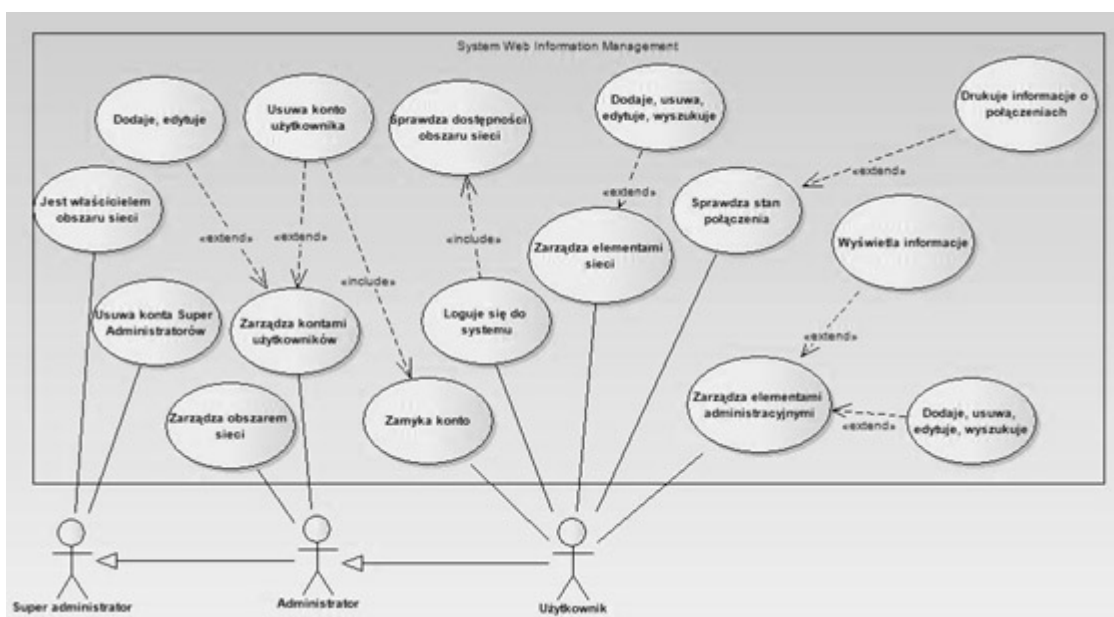
W tym rozdziale można zapoznać się z wewnętrzną specyfikacją tworzonego oprogramowania. Przedstawiono w nim diagram przypadków użycia, który pozwala przeanalizować, jakie zadania są przydzielane poszczególnym aktorom w systemie. Omówiono również model danych – tabele oraz procedury stworzone z wykorzystaniem języka *SQL*. Zaprezentowano także kilka ważnych algorytmów, wykorzystanych w zadaniu. Opisano ważniejsze klasy, które są stworzone w systemie i ich składowe. Na koniec tego rozdziału opisano również, jak jest w systemie realizowana kontrola dostępu i jakie są poziomy dostępu dla różnych użytkowników.

3.1. Diagram przypadków użycia

Diagram, który jest przedstawiony na rysunku 3.1, został wykonany przy użyciu narzędzia Enterprise Architect 7.1. Prezentuje on przypadki użycia dla systemu zarządzania informacją o sieci internetowej. W jego skład wchodzi następujący aktorzy:

- super administrator (dziedziczy wszystko, co ma administrator oraz użytkownik),
- administrator (dziedziczy wszystko, co ma użytkownik),
- użytkownik.

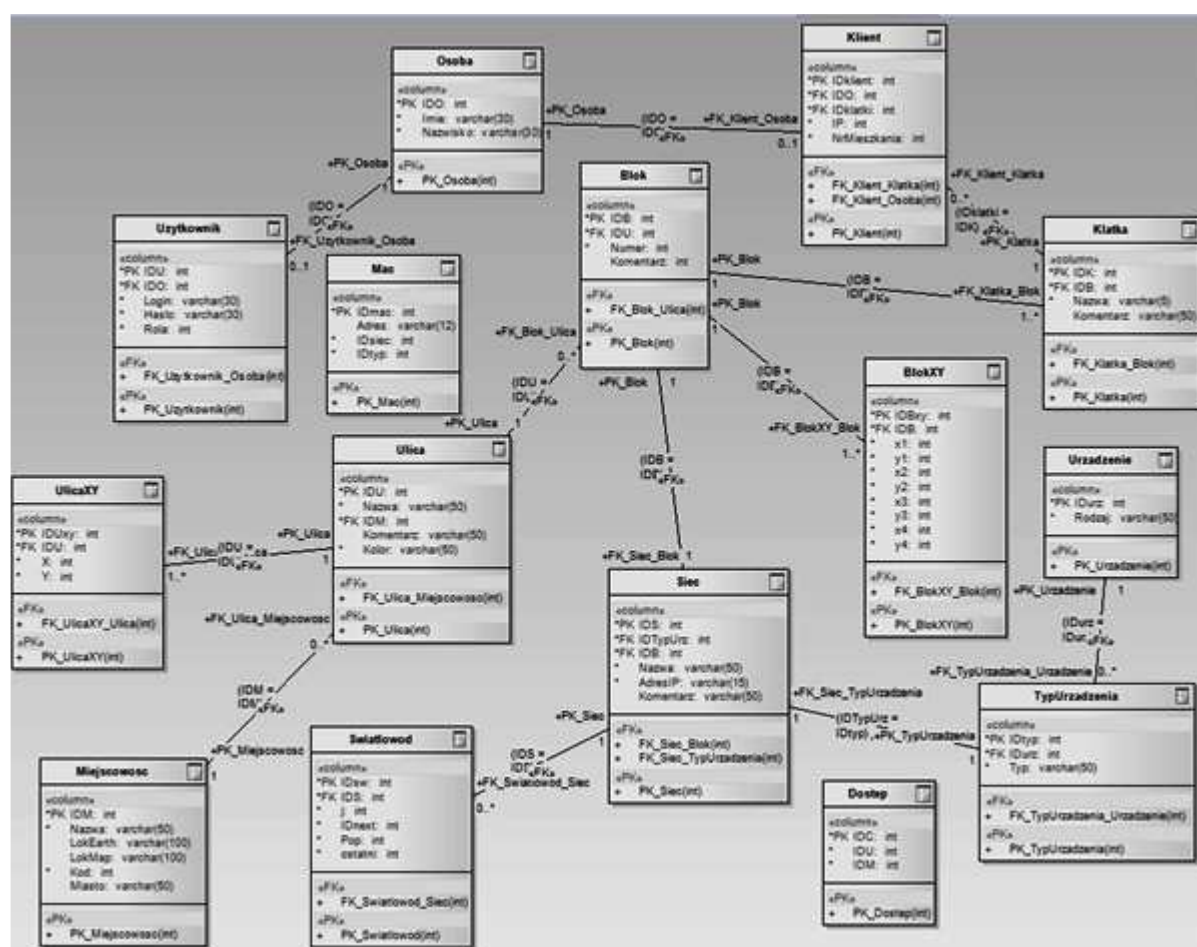
Opis funkcji dostępnych dla użytkowników na różnych poziomach dostępu można znaleźć w podrozdziale 3.6 dotyczącym kontroli dostępu.



Rysunek 3.1. Diagram przypadków użycia

3.2. Model danych

Model danych został wykonany przy użyciu narzędzia Enterprise Architect 7.1. Na rysunku 3.2 zaprezentowano zaprojektowany model. Schemat przedstawia tabele, których opis jest zamieszczony poniżej. Rysunek pokazuje też, jakie są relacje pomiędzy poszczególnymi tabelami, wyszczególnia klucze podstawowe oraz klucze obce w tabelach. Dokładny opis pól tabel można znaleźć w dodatku B niniejszej pracy. Wygenerowany został również raport w formacie HTML, w którym można przejrzeć całą strukturę bazy oraz wszystkie tabele, które ona zawiera. Można też zapoznać się w nim z diagramem przypadków użycia. Ten raport znajduje się wśród załączników do pracy na płycie CD.



Rysunek 3.2. Model danych

Poniżej opisane są w sposób skrócony wszystkie tabele bazy danych.

- Miejscowość

Zawiera informacje o nazwach obszarów sieci (rozumianych jako terytoria, na których działa dana firma, obejmujące miejsca zamieszkania wszystkich jej klientów) w miejscowościach. Każda miejscowość posiada również swoją nazwę oraz kod pocztowy. W tej tabeli jest także

zapisywana lokalizacja plików graficznych z mapami miejscowości wykorzystywanymi w systemie.

- Ulica

Przechowuje podstawowe informacje na temat ulic należących do konkretnych miejscowości. W tabeli są zapisywane między innymi systemowe nazwy kolorów, które są przyporządkowane do ulicy, aby wyświetlić ją potem na mapie terenu.

- UlicaXY

Informacje o współrzędnych poszczególnych odcinków, z których składa się ulica. Dzięki tym współrzędnym może ona potem być rysowana na mapie.

- Blok

Zawiera podstawowe informacje na temat bloków mieszkalnych należących do konkretnych ulic w danej miejscowości. Każdy blok jest rysowany na mapie takim samym kolorem jak ulica, do której należy.

- BlokXY

Przechowuje informacje o współrzędnych czterech punktów (wierzchołków), z których składa się blok mieszkalnych. Dzięki zapamiętaniu w tabeli tych współrzędnych może on potem być rysowany na mapie.

- Klatka

Informacje o klatkach schodowych należących do bloku mieszkalnego. Jeżeli blok nie istnieje, to klatka również nie może istnieć.

- Klient

Zawarte w niej dane pochodzą z tabeli Osoba. Każdy klient jest przydzielony do konkretnej klatki w bloku mieszkalnym na danej ulicy w danej miejscowości.

- Siec

W jednym bloku mieszkalnym może istnieć jedna sieć. Tabela ta zawiera informacje o nazwie sieci, jej adresie IP, o urządzeniu, które pozwala doprowadzić kabel do różnych klientów w bloku mieszkalnym.

- Swiatlowod

Każdy światłowód posiada informację o sieci źródłowej, z której jest doprowadzony, o sieci, do której jest prowadzony, o ilości żył, które się w nim znajdują oraz o tym czy jest to ostatni światłowód czy też znajduje się pomiędzy innymi połączeniami.

- Urządzenie

Spis rodzajów urządzeń, które mogą być użyte do rozdzielania połączenia pomiędzy użytkownikami sieci (klientami).

- TypUrządzenia

Tabela przechowuje informacje o modelach urządzeń sieciowych. Typ urządzenia posiada swój rodzaj. Typ określa konkretny model na przykład koncentratora albo routera.

- Osoba

Zawiera podstawowe informacje (imię i nazwisko) na temat osób, będących użytkownikami systemu jak i klientów. Dane zamieszczone w tabelach Klient oraz Użytkownik są zgodne z danymi w tabeli Osoba.

- Użytkownik

Tabela, która gromadzi dane o nawach użytkowników, hasłach oraz określa poziom dostępu – rolę w systemie zarządzania informacją o sieci internetowej.

- Dostęp

Tabela ta nie jest powiązana z żadną inną. Zawiera tylko i wyłącznie identyfikatory miejscowości oraz użytkowników dzięki czemu jest bardzo przydatna przy określaniu czy dany użytkownik ma dostęp do konkretnego obszaru sieci, czy może się do niego załogować.

- Mac

Tabela nie jest powiązana z żadną inną tabelą w modelu danych. Zawiera identyfikatory sieci oraz urządzeń. Dzięki temu można przydzielać adresy MAC urządzeń w konkretnych sieciach.

3.3. Procedury SQL

W programie Microsoft SQL Server zostało napisanych wiele procedur, które ułatwiają wykonywanie operacji na danych w bazie z użyciem platformy .NET. Niektóre operacje dotyczące danych w bazie dokonywane są poprzez wywołanie tych procedur, a nie listę kolejnych zapytań SQL w aplikacji. Najczęściej służą one do dodawania nowych elementów, edycji już istniejących albo ich usuwania. W poniższej tabeli zaprezentowano trzy przykłady takich procedur. Opis pozostałych można znaleźć w dodatku C niniejszej pracy. Implementacja tych procedur znajduje się skrypcie tworzącym bazę danych, którego fragment znajduje się w dodatku A, natomiast pełny tekst tego skryptu można znaleźć na płycie CD.

| Nazwa procedury | Opis | Parametry |
|-------------------------|--|--|
| DodajBlok | Dodawanie nowego bloku. Parametry określają do jakiej ulicy przyporządkowywany jest blok, jego numer i opcjonalnie komentarz. Identyfikator sieci początkowo ma wartość 0, ale potem podczas tworzenia sieci dla mieszkańców danego bloku ta wartość zostaje zmieniona w innej procedurze. | IDB – identyfikator bloku (wyjście) IDU - identyfikator ulicy (wyjście, wejście) IDS – identyfikator sieci (wyjście, wejście) Numer, Komentarz - (wejścia). |
| EdytujSwiatlowod | Edytowanie sieci polegające na zmianie ilości żył w światłowodzie dochodzącym do danej sieci. | Pop – identyfikator sieci określający sieć źródłową dla danego światłowodu (wejście) j – nowa ilość żył dla danego światłowodu (wejście). |
| UsunUlica | Usuwanie ulicy wraz z informacjami o wszystkich wierzchołkach do niej należących, które określają położenie ulicy na mapie terenu. | IDU – identyfikator ulicy, która ma zostać usunięta (wejście, wyjście). |

3.4. Klasy

W języku *C#* wszystko jest oparte na klasach. Nawet program jest klasą. Podczas implementacji systemu tworzonego w ramach niniejszej pracy oprócz wykorzystania biblioteki klas bazowych (gotowych klas systemowych), zaimplementowano również wiele klas, które można podzielić na grupy ze względu na pełnione przez nie funkcje. Poniżej opisano kilka klas z najważniejszych grup, jakimi są formularze oraz klasy funkcjonalne.

Klasy dziedziczące po *Form* (okienka)

Część zaimplementowanych klas dziedziczy po systemowej klasie *Form*. Są to formularze służące do komunikacji z użytkownikiem. Każdy formularz może zawierać kontrolki takie jak klasy przycisków, paneli, pól tekstowych oraz list. Do najważniejszych klas należą:

- **Database**

Klasa ta udostępnia formularz (okienko, przyciski, pola tekstowe), która pozwala na komunikację z użytkownikiem w celu wygenerowania bazy danych za pomocą pliku

.mdf. Pozwala też na stworzenie nowego obszaru sieci wraz z jego właścicielem. Znajduje zastosowanie w aplikacji *BaseGenerator*.

Ważniejsze metody:

- `private void createButton_Click(object sender, EventArgs e)`

Reakcja na zdarzenie wyboru opcji “Create” w sekcji *Database*. Sprawdzana jest poprawność wprowadzonych przez użytkownika danych związanych z bazą, która ma zostać utworzona. Następuje próba otworzenia połączenia (wykorzystano model bezpośredni połączenia z bazą danych) i zrealizowania polecenia SQL, które powoduje stworzenie bazy danych o nazwie „sieć” (z której korzysta system zarządzania informacją o sieci internetowej) na podstawie istniejącego i wybranego przez użytkownika pliku bazy .mdf.

- `private void accountButton_Click(object sender, EventArgs e)`

Reakcja na zdarzenie wyboru opcji “Create” w sekcji *Account*. Tworzony jest nowy obszar sieci. Dodawany jest również jej właściciel. Na samym początku sprawdzana jest poprawność danych tekstowych, które wprowadził użytkownik.

- **Login**

Klasa, która udostępnia formatkę, pozwalającą na komunikację z użytkownikiem w celu bezpiecznego zalogowania do systemu zarządzania informacją o sieci internetowej. Znajduje zastosowanie w aplikacji *WebManagement*.

Ważniejsze metody:

- `private void logujButton_Click(object sender, EventArgs e)`

Reakcja na zdarzenie wyboru opcji “Log on”. Ta metoda sprawdza, czy istnieje użytkownik o podanej nazwie i hasle, oraz czy ma dostęp do wybranego obszaru sieci. Zapisuje do zmiennych takie wartości jak: identyfikator aktualnego użytkownika, identyfikator aktualnego obszaru sieci, poziom dostępu aktualnie zalogowanego użytkownika.

- `protected override Boolean ProcessCmdKey(ref Message msg, Keys keyData)`

Reakcja na zdarzenie wyboru przycisku *Enter* na klawiaturze. Ta metoda ma podobne działanie do wyżej opisanej.

Klasy funkcjonalne

W systemie zostało zaimplementowanych wiele klas, pogrupowanych ze względu na różne funkcje, które pełnią. Do najważniejszych grup funkcyjnych należą: sprawdzanie

poprawności wprowadzanych danych, dodawanie, edycja oraz usuwanie elementów, testowanie, rysowanie, drukowanie, reakcje na zdarzenia, wyszukiwanie oraz tymczasowy zapis danych. Poniżej opisano najważniejsze z tych klas z uwzględnieniem podziału na funkcje.

- **Sprawdzanie poprawności wprowadzanych danych - *CheckFormat***

Klasa zawierająca metody sprawdzające poprawność danych wpisywanych przez użytkowników systemu. Z tej klasy korzystają obie aplikacje w systemie – *WebManagement* oraz *BaseGenerator*.

Ważniejsze metody:

- `public List<int> checkAccountFormat(string login, string passwd, string name, string surname, string net, string city, string code1, string code2)`

Sprawdzanie poprawności formatu tekstu podczas dodawania nowego konta super administratora, właściciela bazy. Wejście: sześć parametrów typu `string`: `login` (nazwa użytkownika), `passwd` (hasło), `name` (imię), `surname` (nazwisko), `net` (nazwa obszaru sieci), `city` (nazwa miasta), `code1` (pierwsza część kodu pocztowego), `code2` (druga część kodu pocztowego), które są potrzebne aby stworzyć użytkownika i nowy obszar sieci. Wyjście: `List<int>` zwracana jest kolekcja typu `int`, która pozwala wydobyc informacje na temat tego, które pole zostało błędnie wypełnione przez użytkownika.

- `public List<int> checkDatabaseFormat(string nameBase, string server)`

Sprawdzanie poprawności formatu tekstu podczas tworzenia nowej bazy danych. Wejście: dwa parametry typu `string`: `nameBase` (nazwa bazy), `server` (nazwa serwera gdzie powstaje baza), które są potrzebne aby stworzyć nową bazę danych. Wyjście: `List<int>` zwracana jest kolekcja typu `int`, która pozwala wydobyc informacje na temat tego, które pole zostało błędnie wypełnione przez użytkownika.

- **Dodawanie, edycja, usuwanie elementów - *Block***

Klasa ta zawiera zestaw metod, które pozwalają modyfikować zawartość bazy danych – a dokładnie tabel, które przechowują dane o blokach mieszkalnych (dodawanie, usuwanie). Znajduje zastosowanie w aplikacji *WebManagement*.

Ważniejsze metody:

- `public void loadBlockInfo()`

Ładowanie danych z bazy o wybranym bloku do obiektów w panelu informacyjnym dotyczącym bloków mieszkalnych.

- `public void deleteBlock()`

Usuwanie pojedynczego bloku mieszkalnego z bazy danych oraz wszystkich wierzchołków z nim związanych.

- `public void addBlock()`

Inicjalizacja operacji dodawania nowego bloku mieszkalnego do bazy danych.

- `public void newBlock(object sender, MouseEventArgs e)`

Reakcja na zdarzenie używania przycisków myszy w obszarze mapy. Użytkownik wybiera cztery punkty na mapie. Metoda jest odpowiedzialna za to, aby po wyborze każdego wierzchołka nastąpiło sprawdzenie, czy wybrane położenie jest zgodne z algorytmem. Kod źródłowy tej metody znajduje się w dodatku D1 niniejszej pracy.

- **Testowanie - *TestNetwork***

Klasa korzysta ze zdefiniowanych klas systemowych. Służy do sprawdzania stanu połączenia konkretnych sieci oraz poszczególnych klientów firmy z Internetem. Znajduje zastosowanie w aplikacji *WebManagement*.

Ważniejsze metody:

- `public int ping_send(string adresIp)`

Wysłanie polecenia ping oraz oczekiwanie na odpowiedź od adresu IP. Wejście: `string adresIp` – adres IP lub nazwa hosta, do którego zostaje wysłany ping. Wyjście: `int`. Zwracana jest wartość 0 gdy brak odpowiedzi od danego hosta. Wartość 1 zwracana jest gdy połączenie zostało nawiązane i otrzymano odpowiedź.

- **Rysowanie - *Streetdraw***

Klasa dostarczające metody, które rysują na mapie terenu (na panelu w programie) ulice znajdujące się w danym obszarze sieci lub odświeżają widok na mapie. Znajduje zastosowanie w aplikacji *WebManagement*.

Pola:

- `static public int remX;`

Odpowiedzialne za zapamiętania ostatniej wartości współrzędnej X rysowanej ulicy na mapie.

- `static public int remY;`

Odpowiedzialne za zapamiętania ostatniej wartości współrzędnej Y rysowanej ulicy na mapie.

Ważniejsze metody:

- `public void drawAllStreets()`

Narysowanie wszystkich ulic na mapie w danym obszarze sieci, do którego użytkownik jest aktualnie zalogowany.

- `public void drawOneStreet(int idu)`

Narysowanie jednej ulicy, wcześniej wybranej przez użytkownika systemu.

Wejście: `int idu` - zmienna określająca identyfikator ulicy, która ma zostać narysowana na mapie.

- `public void refreshStreetDraw()`

Odświeżanie rysowania ulic, gdy wybrano przycisk *Delete* lub *Backspace* w czasie dodawania nowej ulicy tak, by usunąć poprzednią współrzędną ulicy i narysować zaktualizowaną wersję.

- **Drukowanie - *Printing***

Klasa zawierająca metody do przygotowywania podglądu wydruku, do wydruku oraz ładowania danych przygotowywanych do wydruku. Znajduje zastosowanie w aplikacji *WebManagement*.

Ważniejsze metody:

- `public void loadInfoClientGrid(int idClient)`

Wypełnianie złożonego pola tekstowego, w którym znajduje się pełny adres wybranego klienta. Wejście: `int idClient` - identyfikator klienta, którego adres ma być wyświetlony.

- `public void preparePrint(object sender, System.Drawing.Printing.PrintPageEventArgs e)`

Przygotowanie wydruku listy klientów

- `public void printDoc(int which)`

Drukowanie dokumentu. Wejście: `int which` - zmienna określająca co ma zostać wydrukowane. 1 – wydrukowana zostanie lista klientów. 2 – wydrukowana zostanie lista klientów niepołączonych z siecią.

- **Reakcje na zdarzenia - *ButtonClick***

Klasa, która ma za zadanie dostarczyć metody odpowiedzialne za reakcje na wybieranie niektórych przycisków na klawiaturze w czasie anulowania dodawania nowej ulicy i nowego bloku mieszkalnego. Znajduje zastosowanie w aplikacji *WebManagement*.

Ważniejsze metody:

- `public void OnPressKey(int tribe)`

Reakcja na zdarzenie wyboru przycisków *Delete* lub *Backspace* na klawiaturze w czasie, gdy dodawany jest nowy blok mieszkalny lub nowa ulica w celu wycofania operacji dodawania. Wejście: `int tribe` - zmienna, która pozwala określić jaki element jest aktualnie dodawany, żeby anulować dodawanie konkretnego elementu w bazie danych. Jeśli ma wartość 1 to wtedy anulowane jest dodawanie ulicy. Jeśli ma wartość 2 to wtedy anulowane jest dodawanie bloku mieszkalnego.

- **Wyszukiwanie - *Searcher***

Klasa zawierająca metody odpowiedzialne za wyszukiwanie danych w bazie. Znalezione dane są również wyświetlane. Znajduje zastosowanie w aplikacji *WebManagement*.

Metoda:

- `public void startSearching(int wybor, string fraza)`

Ta metoda przeprowadza wyszukiwanie w bazie danych oraz zajmuje się inicjalizacją wyszukiwania. Wejście: `int wybor` - określa jaki typ elementu jest wyszukiwany.

`string fraza` - fraza tekstu według której przeszukiwana jest baza danych.

- **Klasy do zapisu tymczasowych danych**

Zestaw klas w systemie, które mają bardzo podobną budowę. Każda z nich służy do tymczasowego zapamiętania specyficznych elementów wyciągniętych z bazy danych. Najczęściej jest to para danych: identyfikator oraz nazwa elementu. Obiekty tych klas zawsze są tworzone jako generyczne kolekcje typu `BindingList<NazwaKlasy>` na przykład `BindingList<Admins>`. Taka lista pozwala przechowywać zawsze zaktualizowane dane w obiektach typu `ComboBox` i innych. Oprócz tego, pozwala w szybki sposób identyfikować elementy w bazie danych, za pomocą aktualnie stworzonego obiektu którejś z powyższych klas.

3.5. Wybrane algorytmy

W poniższym podrozdziale opisano niektóre algorytmy, wykorzystywane w programie. Są to przeważnie takie algorytmy, które były trudne w implementacji lub takie, które okazały się szczególnie istotne i warte opisanie. Należy do nich między innymi algorytm ustalania wierzchołków bloku mieszkalnego, tworzenia bazy danych, tworzenia połączenia sieciowego,

testowania połączenia danego numeru IP z Internetem i ładowania danych z bazy do źródła danych obiektu.

Ustalanie wierzchołków bloku mieszkalnego

Algorytm jest wykorzystywany podczas dodawania nowych bloków mieszkalnych. W czasie rysowania bloku, polegającego na wyborze jego wierzchołków każdy wierzchołek jest sprawdzany pod kątem tego, czy odpowiada wymaganiom, które przedstawiono w algorytmie. Następuje analiza, czy dany wierzchołek może być zatwierdzony i czy można przejść do wyboru kolejnego wierzchołka (jeśli to nie jest ostatni – czwarty). Zakładając, że:

X_0, Y_0 – współrzędne pierwszego wierzchołka,

X_1, Y_1 – współrzędne drugiego wierzchołka,

X_2, Y_2 – współrzędne trzeciego wierzchołka,

X_3, Y_3 – współrzędne czwartego wierzchołka,

algorytm można opisać poniższymi punktami.

1. Pierwsza współrzędna (X_0, Y_0) nie podlega analizie. Może być dobrana dowolnie.
2. Druga współrzędna powinna być dobrana ze spełnieniem poniższych warunków:
 - $X_1 > X_0$,
 - $Y_1 > Y_0$.
3. Trzecia współrzędna powinna być dobrana ze spełnieniem poniższych warunków:
 - $X_1 > X_2$
 - $Y_2 > Y_1$
4. Czwarta współrzędna powinna być dobrana ze spełnieniem poniższych warunków:
 - $X_3 < X_0$
 - $X_3 < X_2$
 - $Y_3 < Y_2$
 - $Y_3 < Y_0$

Wybranie współrzędnych według powyższego algorytmu pozwala na późniejsze poprawne funkcjonowanie obliczania układów równań w celu wykrywania wyboru konkretnego bloku mieszkalnego przez użytkownika. Kod źródłowy algorytmu zamieszczono w dodatku D1.

Tworzenie bazy danych

Algorytm służy do stworzenia na serwerze lokalnym bazy danych na podstawie istniejącego pliku *.mdf* oraz parametrów podanych przez użytkownika. Algorytm można opisać poniższymi punktami, a jego kod źródłowy znajduje się w dodatku D2.

1. Tworzenie ciągu znaków połączenia z bazą danych(`string conStr`).
2. Tworzenie nowego obiektu połączenia `SqlConnection`. Do jego konstruktora przekazywany jest stworzony wcześniej ciąg znaków połączenia.
3. Próba otwarcia połączenia z bazą danych.
4. Utworzenie obiektu polecenia `SqlCommand`.
5. Przypisanie ciągu znaków do polecenia SQL, który wykonuje próbę utworzenia bazy danych na podstawie istniejącego pliku

Tworzenie połączenia sieciowego

Algorytm ten jest najbardziej skomplikowanym w tworzonym oprogramowaniu. Ze względu na bardzo duży stopień skomplikowania będzie on opisany w znacznym uproszczeniu. Użytkownik dodając nowe połączenie pomiędzy istniejącymi sieciami, ma do wyboru tylko sieci, pomiędzy którymi połączenie w ogóle jest możliwe. Zaimplementowano szybką aktualizację wybranych sieci (za pomocą kolekcji generycznych), która pozwala ustrzec się błędów takich, jak na przykład wybór tej samej sieci jako źródło i cel, albo sieć pośrednia (kod źródłowy zamieszczono w dodatku D.3). Sam algorytm tworzenia połączenia, po zbadaniu poprawności wprowadzanych danych, można opisać poniższymi punktami.

1. Sprawdzenie czy ilość żył w światłowodzie wystarczy, aby poprowadzić wybrane połączenie między sieciami. Jeśli żył jest za mało to dodawanie połączenia jest anulowane.
2. Sprawdzenie czy od wybranej sieci źródłowej nie odchodzą już jakieś inne światłowody. Jeśli nie ma innych rozgałęzień, należy przejść do punktu 3 niniejszego algorytmu. W przeciwnym wypadku należy przejść do punktu 8.
3. Sprawdzenie jaka sieć jest źródłem dla sieci, która została przed chwilą wybrana jako źródło. Jeżeli nie ma poprzednika operacja zostaje zakończona a użytkownik jest powiadamiany o braku możliwości dodania połączenia.
4. Zbadanie czy źródłowa sieć posiada wystarczającą ilość żył na wyjściu światłowodu. Jeśli nie, operacja zostaje zakończona a użytkownik jest powiadamiany o braku możliwości dodania połączenia..

5. Dla każdego połączenia pomiędzy poszczególnymi sieciami dodawana jest do bazy informacja o nowym światłowodzie. Parametr "ostatni" ma wartość '0' dla pośrednich sieci a '1' dla sieci docelowej.
6. Uaktualnienie ilość kabli na wyjściu sieci.
7. Aktualna sieć połączona ze źródłem sama staje się źródłem. Powrót do punktu 5.
8. Sprawdzenie czy wystarczy żył na połączenia w światłowodzie
9. Wybranie światłowodów, które są końcowe (nie mają następników).
10. Sprawdzenie który z wybranych światłowodów prowadzi do sieci, która została wybrana jako źródło.
11. Zapamiętanie parametrów znalezionej światłowodu (ilość żył, sieć źródłowa, poprzednik).
12. Ponowne zbadanie czy na podstawie otrzymanych wyników jest możliwe poprowadzenie połączenia ze względu na ilość żył w światłowodzie.
13. Edytowanie ścieżki światłowodów prowadzących od sieci źródłowej do sieci, do której prowadzi wybrany końcowy światłowod.
14. Nowy przydział żył dla światłowodów wcześniej istniejących oraz dla nowych.
15. Dodanie do bazy danych informacji o nowych światłowodach.

Testowanie połączenia numeru IP z Internetem

Ta operacja jest realizowana za pomocą wysyłania polecenia `ping` do komputera klienta. Algorytm można opisać poniższymi punktami.

1. Wybór klienta.
2. Stworzenie nowego obiektu klasy `IPAddress` (jest to klasa systemowa z biblioteki klas platformy). Za jej pomocą tworzony jest adres IP lub hostname za pomocą ciągu znaków podanych przez użytkownika.
3. Stworzenie nowych obiektów klas `Ping` oraz `PingReply`.
4. Ustalenie granicznej wartości czasu (*timeout*).
5. Skorzystanie z gotowej funkcji `send(adresIP, timeout)` do wysłania `ping`.
6. Sprawdzenie czy połączenie zostało poprawnie nawiązane za pomocą właściwości `IPStatus.Success`.

Ładowanie danych z bazy do źródła danych obiektu

W modelu bezpołączeniowym bazy danych obiekt klasy `DataSet` używany jest jako pamięć podręczna. Obiekt `DataAdapter` służy jako pośrednik między obiektem `DataSet` a źródłem

danych, z którego pochodzą dane w pamięci. Poniższy algorytm jest nieraz wykorzystywany w programie do tego, aby załadować dane znajdujące się w bazie danych do pamięci podręcznej. Zmodyfikowane dane w pamięci podręcznej są następnie aktualizowane w bazie danych. Algorytm można opisać poniższymi punktami, a jego kod źródłowy zamieszczono w dodatku D.4.

1. Utworzenie nowego obiektu klasy `DataSet`.
2. Ustalenie treści polecenia SQL, żeby określić z jakich tabel należy pobrać dane.
3. Ustalenie ciągu znaków połączenia z bazą danych (tak zwany *connection string*).
4. Tworzenie nowego obiektu połączenia `SqlConnection`. Do jego konstruktora przekazywany jest stworzony ciąg znaków połączenia.
5. Tworzenie nowego obiektu `SqlDataAdapter` i wypełnienie za jego pomocą tabeli z danymi z bazy danych.
6. Tworzenie nowego obiektu klasy `BindingSource`, ustalenie dla niego źródła danych oraz tabeli źródłowej.
7. Przypisanie obiektu stworzonego w punkcie 6 do obiektu, który będzie korzystał z danych (na przykład obiekt `ComboBox`) oraz ustalenie jaki element z bazy będzie wyświetlany (*display member*) a jaki będzie identyfikatorem (*value member*).

3.6. Kontrola dostępu

W systemie zarządzania informacją o sieci internetowej tworzonego dla firmy *CzarNet* użytkownicy korzystają z dostępnych funkcji na trzech różnych poziomach dostępu. Poziom dostępu jest ustawiany podczas tworzenia nowego użytkownika. Nazwy poziomów dostępu to:

- super administrator,
- administrator,
- użytkownik.

Użytkownik to osoba w systemie, która posiada najmniejsze uprawnienia. Użytkownik ma dostęp do wielu funkcji systemu takich jak dodawanie, usuwanie i edytowanie obiektów administracyjnych oraz sieciowych. Nie ma dostępu do funkcji administratorów i super administratorów. Sam nie może usuwać żadnego rodzaju konta w systemie.

Administrator to użytkownik w systemie, który posiada prawie pełną funkcjonalność. Może wykonywać wszystkie czynności, które wykonuje użytkownik. Może również usuwać

konta innym użytkownikom. Nie może korzystać jedynie z funkcji znajdującego się wyżej od niego w hierarchii super administratora

Super administrator to użytkownik w systemie o największych uprawnieniach. Korzysta ze wszystkich dostępnych funkcji. Oprócz funkcji, do których mają dostęp zwykli użytkownicy i administratorzy, super administrator może usuwać konta innym super administratorom. Poza tym jest właścicielem obszaru sieci w danej miejscowości. Jego konto jest tworzone podczas tworzenia nowego obszaru sieci. Tylko on może też usuwać całkowicie obszar sieci i swoje konto.

Podczas logowania do systemu następuje kontrola dostępu. Sprawdzane jest czy istnieje użytkownik o podanej nazwie, który identyfikowany jest przez podane hasło. W momencie logowania użytkownik wybiera, do jakiego istniejącego obszaru sieci chce się zalogować. Jeśli stwierdzone zostanie, że użytkownik istnieje w systemie, wtedy jest sprawdzane czy dany użytkownik ma prawo dostępu do wybranego obszaru sieci. Jeśli nie, nie będzie mógł się zalogować.

4. Specyfikacja zewnętrzna

Niniejszy rozdział poświęcony jest opisowi specyfikacji zewnętrznej tworzonego oprogramowania. Omówiono w nim między innymi wymagania sprzętowe oraz programowe powstałego systemu. Przedstawiono również kilka wskazówek dla administratora, który będzie rozpoczynał pracę z oprogramowaniem. Opisano jak przebiega instalacja, a także zaprezentowano opis menu oraz zakładek dostępnych w programie.

4.1. Wymagania sprzętowe oraz programowe

Do sprawnego działania systemu zarządzania informacją o sieci internetowej wystarcza procesor Intel Celeron 1.5GHz i pamięć RAM 196MB. Są to minimalne wymagania sprzętowe, choć autor zdecydowanie poleca wykorzystanie oprogramowanie na lepszym sprzęcie, tak by praca była wygodna i nie zbyt czasochłonna.

System wymaga do swojego działania kilku programów, bez których nie da się z niego korzystać. Do tych programów należą:

- SQL Server 2005,
- SQL Server 2005 Management Studio,
- .NET Framework 3.5.

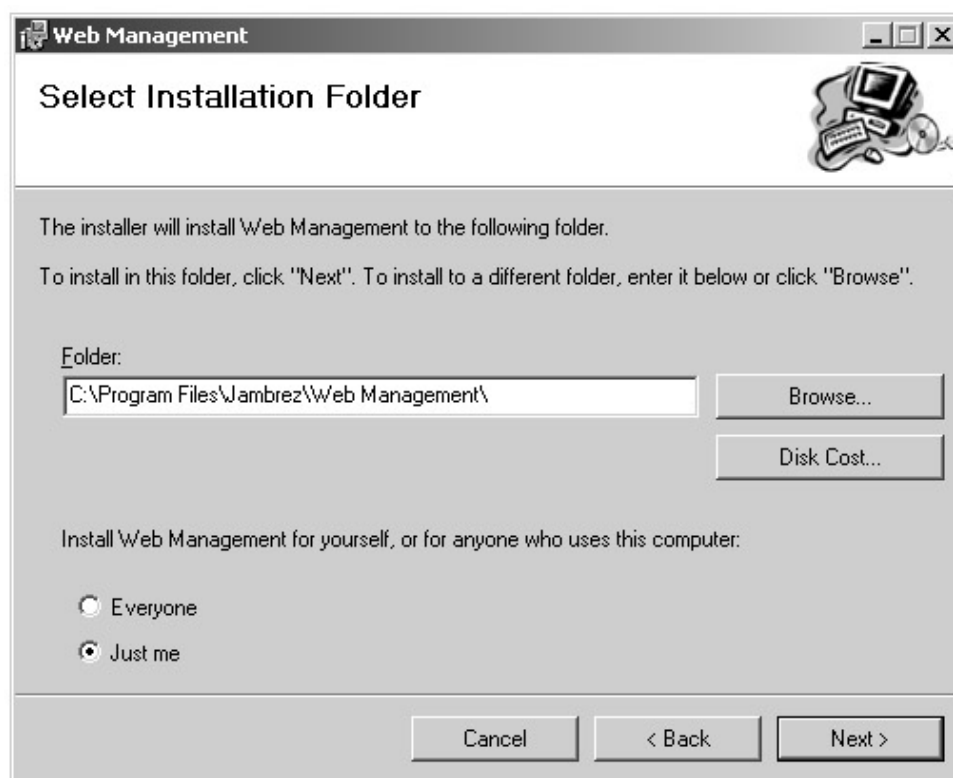
System składa się bazy danych na serwerze SQL Server oraz dwóch aplikacji - *WebManagement*, która umożliwia korzystanie z funkcji dostępnych dla stworzonych obszarów sieci oraz *BaseGenerator*, która służy do stworzenia bazy danych oraz nowego obszaru sieci wraz z jego właścicielem. Zanim rozpocznie się korzystanie z systemu niezbędne jest stworzenie bazy danych na komputerze. W tym celu należy uruchomić aplikację *BaseGenerator*, która wygeneruje bazę na podstawie pliku *siec.mdf* na serwerze lokalnym. Aby użyć innego serwera niż lokalny należy wykonać na nim skrypt tworzący bazę danych. Należy również założyć konto administratora za pomocą tej samej aplikacji oraz stworzyć nowy obszar sieci tak, by móc zacząć korzystanie z systemu. W katalogu, gdzie zainstalowany jest system można znaleźć dwa pliki konfiguracyjne o nazwach *WebManagement* oraz *BaseGenerator*. Należy w tych plikach wpisać w sekcji *Server* nazwę serwera, z którym będą łączyły się aplikacje.

4.2. Opis instalacji

Aby zainstalować system na komputerze należy wybrać plik *Setup.exe* (załączony do pracy na płycie CD), który uruchomi instalator. Zainstalowane zostaną aplikacje

WebManagement oraz *Basegenerator*, które będą umieszczone w wybranym przez użytkownika katalogu. W tym katalogu znajdą się również dwa pliki konfiguracyjne. Instalacja przebiega w sposób typowy dla tego typu oprogramowania. Jeżeli na komputerze, na którym instalowane jest oprogramowanie nie ma programu .NET Framework 3.5 instalacja zostanie przerwana do momentu jego pobrania i zainstalowania. Podczas instalacji pojawia się kilka okienek służących do komunikacji z użytkownikiem:

- okno powitalne,
- wybór lokalizacji na dysku oraz określenie, którzy użytkownicy komputera mają mieć dostęp do programu (patrz rysunek 4.1),
- rozpoczęcie instalacji oraz jej zakończenie.



Rysunek 4.1. Instalacja

4.3. Opis działania systemu

Użytkownik ma do dyspozycji dwie różne aplikacje, z których każda jest przeznaczona do odrębnego zastosowania. *BaseGenerator.exe* służy do stworzenia bazy danych oraz nowego obszaru sieci wraz z jego właścicielem. Natomiast *WebManagement.exe* to aplikacja główna, która umożliwia korzystanie z funkcji dostępnych dla stworzonych obszarów sieci. Poniżej opisano zakładki i różne opcje wyboru możliwe dla tych aplikacji.

Aplikacja *BaseGenerator.exe*

Po wyborze aplikacji *BaseGenerator.exe* pojawia się okno, które pokazano na rysunku 4.2. Ta aplikacja służy do dwóch celów. Pozwala stworzyć bazę danych (wtedy jest widoczny panel jak na rysunku 4.2) oraz konto użytkownika i obszar sieci(rysunek 4.3).

Aby stworzyć bazę danych na serwerze lokalnym za pomocą istniejącego pliku *siec.mdf* należy w menu *File* wybrać opcję *Create Database*. Wybór przycisku *Load from file* powoduje otwarcie okna dialogowego, za pomocą którego wybiera się plik z bazą danych. W jednym z pól tekstowych (tylko do odczytu) pojawi się nazwa wybranego pliku. Użycie przycisku *Create* sprawia, że nastąpi próba stworzenia na serwerze lokalnym bazy o nazwie *siec*. Jeżeli nie jest możliwe utworzenie połączenia, został wybrany niewłaściwy plik, wpisano niepoprawne dane odnośnie do serwera lub wystąpiły inne problemy to wyświetlony zostanie odpowiedni komunikat i operacja zostanie anulowana.



Rysunek 4.2. Tworzenie bazy danych

W zakładce menu *File* możliwa jest również do wyboru opcja *Create Account*, która pozwala stworzyć konto użytkownika. Należy wypełnić wszystkie pola tekstowe danymi a następnie wybrać opcję *Create*, by utworzyć nowe konto. Gdy któreś pola nie zostaną wypełnione lub będą wypełnione błędnie, pojawi się komunikat i konieczne będzie dokonanie zmian. Jeżeli konto zostanie poprawnie utworzone, użytkownik zostanie o tym poinformowany za pomocą komunikatu w polu statusu (jak to widać rysunku 4.3). Użytkownik, do którego przypisano prawo posiadania obszaru sieci, będzie mógł logować się

do systemu na poziomie uprawnień super administratora. Tylko on będzie mógł usunąć stworzony obszar sieci, którego jest właścicielem.



Rysunek 4.3. Tworzenie obszaru sieci i jej właściciela

Aplikacja WebManagement.exe

Okienko przedstawione na rysunku 4.4 pojawia się podczas uruchomienia aplikacji *WebManagement*. W odpowiednie pola użytkownik powinien wpisać nazwę i hasło oraz wybrać z listy obszar sieci, do którego chce się zalogować. Zostanie zalogowany tylko wtedy, gdy posiada uprawnienia dostępu do konkretnej sieci. Po wpisaniu błędnych danych pojawia się komunikat o błędzie. Jeżeli użytkownik nie posiada uprawnień do danego obszaru, to również zostanie wyświetlony odpowiedni komunikat. Wybranie na klawiaturze przycisku *Enter* działa tak jak wybór opcji Log on – powoduje zalogowanie do systemu.



Rysunek 4.4. Logowanie do systemu

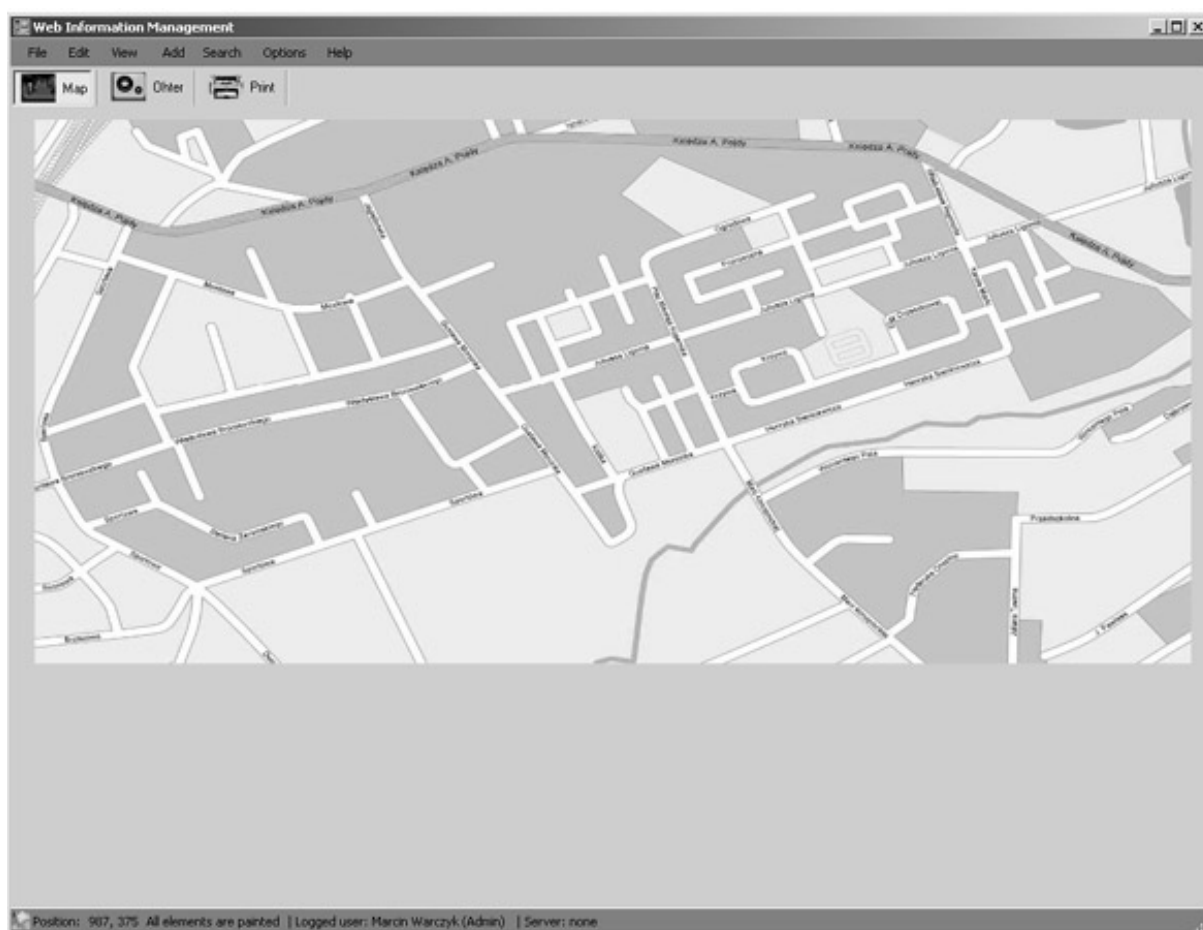
Na rysunku 4.5 przedstawiono wygląd okna głównego aplikacji *WebManagement*. Okno to staje się widoczne dopiero po poprawnym zalogowaniu użytkownika do systemu. Użytkownik ma do dyspozycji szereg opcji i zakładek, które zostały opisane poniżej.

Panel statusu w dolnej części aplikacji zawiera wiele ważnych informacji wyświetlanych w następującej kolejności:

- pozycja wskaźnika myszy na panelu z mapą,
- aktualnie wykonywana czynność,
- imię i nazwisko aktualnie zalogowanego użytkownika oraz nazwa poziomu dostępu,
- lokalizacja serwera.

Po wyborze prawego przycisku myszy w obszarze panelu z mapą pojawia się menu z możliwością wyboru kilku opcji odświeżania rysunku:

- odświeżanie mapy (*Refresh maps*),
- rysowanie mapy (*Draw map*),
- rysowanie zdjęcia satelitarnego (*Draw earth view*),
- rysowanie bez mapy, tylko tło (*Draw no map*).



Rysunek 4.5. Główne okno aplikacji

Poniżej opisano szczegółowo dostępne zakładki menu aplikacji *Web Management*.

- **File**

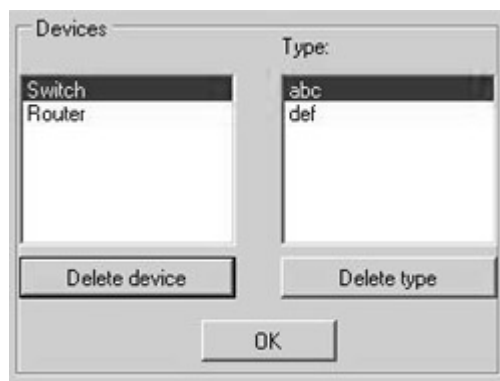
Exit - wybranie tej opcji powoduje zamknięcie aplikacji, nie pojawia się ponownie okno logowania.

Log off - powoduje wylogowanie użytkownika, zamykane jest główne okno aplikacji; pojawia się okno logowania (rysunek 4.4) z pustymi polami do ponownego zalogowania.

Print - wybór tej opcji powoduje otwarcie zakładki pomocniczej o nazwie *Print*, która udostępnia kilka opcji wydrukowania listy różnych klientów; można też zobaczyć podgląd wydruku bez samego drukowania.

- **Edit**

Network devices - wybór tej opcji w programie powoduje wyświetlenie panelu, który jest przedstawiony na rysunku 4.6. Na liście po lewej stronie rysunku 4.6 widać spis rodzajów dostępnych urządzeń sieciowych. Po prawej stronie na liście znajdują się konkretne modele urządzeń danego rodzaju. Wybór opcji *Delete Type* powoduje usunięcie informacji o wybranym modelu urządzenia z bazy danych. Wybranie *Delete device* sprawi, że zostanie usunięta z bazy danych informacja o rodzaju urządzenia wraz ze wszystkimi dziedziczącymi po nim modelami. Wybór opcji *OK* sprawi, że panel zostanie zamknięty.



Rysunek 4.6. Usuwanie urządzeń sieciowych

- **View**

Streets - następuje przejście na zakładkę *Map* (opisaną poniżej), na której znajduje się panel z mapą. Na tym panelu rysowane są wszystkie ulice należące do danego obszaru sieci według koloru wcześniej przyporządkowanego przez użytkownika dla każdej z nich.

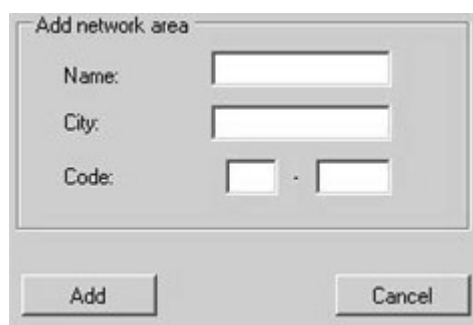
Blocks - następuje automatyczne przejście na zakładkę *Map*, na której znajduje się panel z mapą. Na panelu rysowane są wszystkie bloki mieszkalne należące do danego obszaru sieci według koloru wcześniej przyporządkowanego do ulic przez użytkownika.

Optical fibres - następuje przejście na zakładkę *Map*, na której znajduje się panel z mapą. Na panelu rysowane są wszystkie połączenia światłowodowe należące do danego obszaru sieci. Połączenia pokazywane są w sposób logiczny a nie według ściśle geograficznego położenia.

All - następuje przejście na zakładkę *Map*, na której znajduje się panel z mapą. Na panelu rysowane są wszystkie ulice, bloki mieszkalne oraz logiczne połączenia pomiędzy sieciami (wszystkie te elementy należą do aktualnego obszaru sieci).

- **Add**

Network Area - wybranie tej zakładki umożliwia dodanie nowego obszaru sieci (jest to dostępne tylko dla administratorów i super administratorów). W pola tekstowe panelu o nazwie *Add network area* (rysunek 4.7) należy wpisać wszystkie dane: nazwę nowego obszaru sieci, nazwę miasta oraz kod miasta. Poprawność wprowadzonych danych zostanie sprawdzona i w razie błędnego wypełnienia pól będzie konieczne ponowne ich wprowadzenie. Wybór opcji *Cancel* powoduje anulowanie dodawania a wybór *Add* spowoduje dodanie informacji o nowym obszarze sieci do bazy danych.



Rysunek 4.7. Tworzenie nowego obszaru sieci

Street - wybór zakładki powoduje pojawienie się panelu *Adding street* (rysunek 4.8). Dodawanie nowej ulicy jest realizowane za pomocą podania przez użytkownika niezbędnych danych do tego potrzebnych. W pola tekstowe panelu należy wpisać dane: nazwę ulicy i komentarz (opcjonalnie). Należy również z rozwijanej listy wybrać kolor, który zostanie przyporządkowany ulicy. Kolorem tym będzie rysowana ulica na mapie. Po wprowadzeniu poprawnych danych należy wybrać *Start painting*, które spowoduje zamknięcie panelu i udostępnienie możliwości zaznaczania na mapie współrzędnych poszczególnych odcinków ulicy. Istnieje możliwość anulowania poprzednio zaznaczonych odcinków poprzez wybranie na klawiaturze *Backspace* lub *Delete*. Gdy któryś z nich zostanie wybrany i jeśli nie zaznaczono do tej pory żadnych odcinków, wtedy cała operacja

dodawania nowej ulicy zostanie wycofana. Gdy użytkownik zaznaczy już wszystkie współrzędne, powinien wybrać opcję *End*, aby poprawnie zakończyć dodawanie.

The image shows a Windows-style dialog box titled "Adding street". It has three main input areas: "Name:" with a text box containing "Mostowa", "Describe:" with a large empty text area, and "Colour:" with a small color swatch and a dropdown menu showing "CornflowerBlue". To the right of these fields are two buttons: "Start painting" and "Cancel".

Rysunek 4.8. Tworzenie nowej ulicy

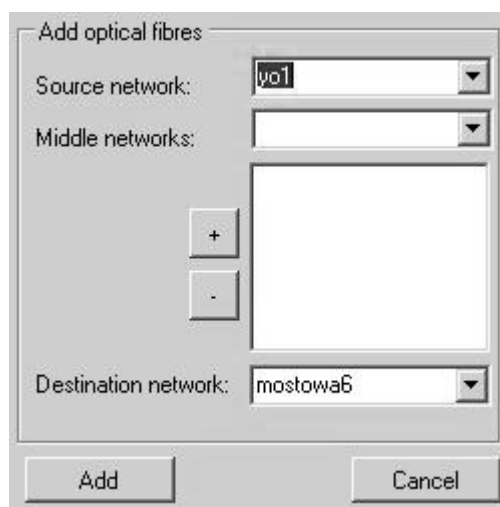
Block of flats - wybór tej zakładki spowoduje pojawienie się panelu *Adding block* (rysunek 4.9). Z listy ulic (*Street*) należy wybrać taką, do której użytkownik chce przyporządkować blok. Należy wybrać numer bloku i ewentualnie wpisać komentarz. Po wyborze opcji *Start painting* na mapie będzie się zaznaczać po kolei cztery wierzchołki, z których będzie składał się blok mieszkalny (opis algorytmu układu wierzchołków można znaleźć w rozdziale poprzednim dotyczącym specyfikacji wewnętrznej w punkcie opisującym wybrane algorytmy). Aby anulować dodawanie bloku należy wybrać na klawiaturze przyciski *Backspace* lub *Delete*.

The image shows a Windows-style dialog box titled "Adding block". It has three main input areas: "Street:" with a dropdown menu showing "mostowa", "Number:" with a spin box showing "2", and "Description:" with a large empty text area. To the right of these fields are two buttons: "Start painting" and "Cancel".

Rysunek 4.9. Tworzenie nowego bloku mieszkalnego

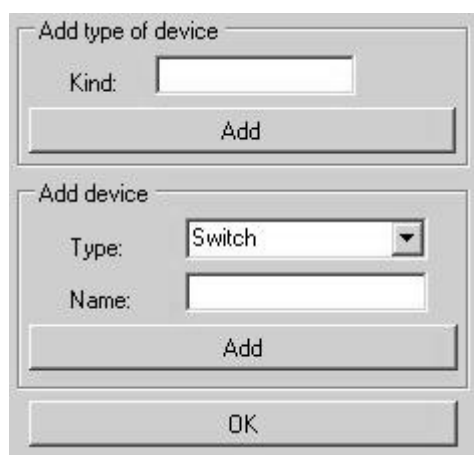
Connection - aby możliwe było dalsze dokonywanie operacji, musi być utworzona serwerownia w aktualnym obszarze sieci. Użytkownik zostanie o tym poinformowany. Jeśli serwerownia istnieje, pojawi się panel jak na rysunku 4.10. Aby powstało połączenie muszą istnieć przynajmniej dwie sieci. Z listy *Source Network* (rysunek 4.10) należy wybrać sieć docelową, z której będzie poprowadzony światłowód. Po wyborze źródła na liście *Destination network* pojawią się sieci, z którymi można nawiązać połączenie. Po tym

wyborze stanie się dostępna lista *Middle network*. Z niej można wybrać (nie jest to wymagane) sieci pośrednie, przez które ma przebiegać światłowód. Po wyborze takiej pośredniej sieci z listy rozwijalnej należy wybrać opcję „+” aby dodać ten element lub przycisk „-” jeśli trzeba go usunąć z listy sieci pośrednich. Po wyborze wszystkich sieci należy wybrać opcję *Add*. Sprawdzana jest wtedy dostępność światłowodu i możliwość połączenia z siecią na podstawie algorytmu, którego opis można znaleźć w rozdziale poprzednim dotyczącym specyfikacji wewnętrznej w punkcie opisującym wybrane algorytmy



Rysunek 4.10. Tworzenie nowego połączenia pomiędzy sieciami

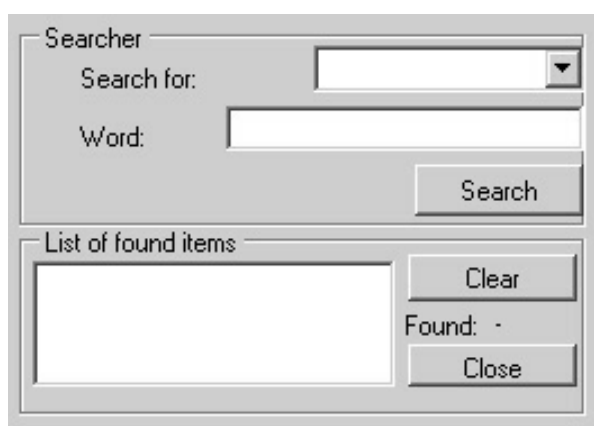
Device - wybór tej zakładki spowoduje pojawienie się panelu *Adding type of device* (rysunek 4.11). Aby dodać nowy rodzaj urządzenia należy w sekcji *Kind* (rysunek 4.11) wpisać nazwę rodzaju nowego urządzenia i wybrać opcję *Add*. Aby dodać nowy model urządzenia należy wybrać z listy *Type* rodzaj urządzenia, następnie wpisać nazwę nowego modelu i wybrać *Add*. Aby zamknąć panel należy nacisnąć *OK*.



Rysunek 4.11. Tworzenie nowych urządzeń sieciowych

- **Search**

Elements - po wybraniu tej opcji pojawia się panel przedstawiony na rysunku 4.12. Istnieje możliwość wyszukiwania klienta, sieci, bloku, ulicy oraz urządzenia na podstawie jego adresu MAC. Z listy *Search for* należy wybrać rodzaj elementu, który użytkownik chce wyszukać. W polu tekstowym *Word* należy wpisać nazwę tego, co ma zostać wyszukane. Wybór opcji *Search* rozpoczyna wyszukiwanie. Ponieważ elementów wyszukanych może być wiele więc wszystkie znalezione wyświetlane są na liście. Po wyborze elementu z listy, wyświetlane są już informacje szczegółowe na jego temat. W przypadku wyszukanego bloku, sieci lub ulicy, po wyborze elementu z listy rysowany jest odpowiedni obiekt na mapie. Opcja *Clear* służy do czyszczenia listy znalezionych elementów a użycie przycisku *Close* spowoduje, że panel wyszukiwania zostanie zamknięty. W polu obok pola *Found* wyświetlana jest ilość znalezionych elementów.



Rysunek 4.12. Panel wyszukiwania

Not connected Network - wybór tej opcji powoduje, że wyszukiwane są wszystkie sieci, które nie są podłączone do Internetu.

- **Options**

Show block Information - wybranie tej zakładki spowoduje, że program od tej chwili będzie rozpoznawał zaznaczone przez użytkownika współrzędne na mapie. Wybranie po raz drugi tej opcji powoduje jej wyłączenie. Jeśli użytkownik wybierze współrzędną (za pomocą myszki) w miejscu, gdzie na mapie jest blok mieszkalny, wtedy też zostanie wyświetlony panel ze wszystkimi informacjami dotyczącymi tego bloku z możliwością dodawania, usuwania i edytowania bloku, klatek schodowych, klientów, sieci. Panel który się pojawi jest przedstawiony na rysunku 4.13.

Blok można usunąć używając przycisku *Delete* w sekcji *Options*. Można też edytować blok wybierając opcję *Edit*. Z kolei wybranie *OK* spowoduje zamknięcie panelu. Jeśli w

bloku istnieje już sieć to przycisk *Add* w sekcji *Network* nie będzie widoczny (tak jak to widać na rysunku 4.13). Użycie przycisku *Add* spowoduje wyświetlenie panelu, który komunikuje się z użytkownikiem i pobiera wszystkie szczegółowe informacje potrzebne do tego, żeby stworzyć sieć w danym bloku mieszkalnym. Wybór przycisku *Edit* powoduje, że można poddać aktualną sieć edycji a używając opcji *Delete* można ją usunąć.

| Block information | | Network | |
|-------------------|-------------------|------------|--|
| City: | Leszczyny Osiedle | Edit | |
| Street: | mostowa | | |
| Number: | 5 | Delete | |
| Staircase count: | 1 | | |
| Network: | mostowa5 | Staircases | |
| Colour: | DarkOrchid | | |
| Cable: 38 | | Add | |
| Options | | View | |
| Delete Edit OK | | | |

Rysunek 4.13. Wyświetlanie informacji o bloku

Sekcja *Staircase* jest odpowiedzialna za dostarczenie informacji o klatkach schodowych w bloku mieszkalnym. Wybierając opcję *Add* można łatwo dodać informację o nowej klatce schodowej. Pojawi się nowy panel, w którym należy wpisać potrzebne do tego dane. Jeśli blok mieszkalny ma przynajmniej jedną klatkę schodową to wybranie *View* spowoduje wyświetlenie informacji na temat istniejących klatek (rysunek 4.14). Wybierając na klawiaturze przyciski strzałek *Right* i *Left* można łatwo uzyskać informacje na temat kolejnych klatek w blokach mieszkalnych. W sekcji *Options* można usuwać klatkę (przycisk *Delete*), edytować ją (przycisk *Edit*) oraz zamknąć panel *Staircase information* (przycisk *OK*). W sekcji *Clients* można w łatwy sposób dodać klienta (wybierając opcję *Add*). Za pomocą przycisku *View* można sprawdzić jacy klienci zamieszkują daną klatkę.

| Staircases information | | Clients | |
|------------------------|---------|---------|--|
| 1/1 | | Add | |
| Name: | a | | |
| Block: | 6 | View | |
| Street: | mostowa | | |
| Clients count: | None | | |
| Options | | | |
| Delete Edit OK | | | |

Rysunek 4.14. Wyświetlanie informacji o klatce schodowej w bloku mieszkalnym

Server location - wybór tej zakładki umożliwia dodanie serwerowni (jest to dostępne tylko dla administratorów i super administratorów). Może istnieć tylko jedna serwerownia na cały obszar sieci. Jeżeli jeszcze nie istnieje to będzie dostępny przycisk *Add*, którego użycie spowoduje stworzenie serwerowni w sieci o wybranej wcześniej z listy istniejącej już nazwie. Należy też w pole *Optical Fiber* wpisać ile żył ma mieć światłowód źródłowy dla serwerowni.

Jeżeli serwerownia już istnieje, będzie wtedy dostępny przycisk *Delete existing*, którego wybór spowoduje usunięcie serwerowni oraz wszystkich sieci i światłowodów w danym obszarze sieci. Użycie przycisku *Cancel* sprawi, że panel serwerowni zostanie zamknięty.

Maps - wybranie tej opcji powoduje, że wyświetlona zostaje aktualna lokalizacja na dysku plików graficznych z mapami terenu (*Map* oraz *EarthView*). Przyciski *Change* pozwalają zmienić lokalizację plików za pomocą okien dialogowych, z których może skorzystać użytkownik.

Accounts - zakładka ta umożliwia zarządzanie kontami użytkowników (jest to dostępne tylko dla administratorów i super administratorów). W zakładce *Add* można dodać konto nowego użytkownika lub administratora (rysunek 4.15). Należy w odpowiednich polach wpisać imię, nazwisko, nazwę użytkownika, hasło oraz zaznaczyć czy nowo rejestrowana osoba będzie administratorem czy zwykłym użytkownikiem. Naciśnięcie *Add* powoduje dodanie użytkownika do systemu. Zostanie on przypisany do danego (aktualnie używanego) obszaru sieci i do niego tylko będzie mógł się logować. Naciśnięcie *Cancel* sprawi, że panel kont użytkowników zostanie zamknięty.

W zakładce *Show* można znaleźć dwie listy: listę administratorów oraz listę użytkowników. Za pomocą przycisku *Delete user* można usunąć konto użytkownika. Korzystając z przycisku *Delete admin* można usunąć konto administratora, ale może to robić tylko super administrator. Zakładka *Super admins* pozwala podejrzeć jacy super administratorzy korzystają z systemu. Nie ma możliwości dokonywania żadnych zmian na ich danych.

Clear all Elements - wybór tej opcji (dostępne tylko dla administratorów i super administratorów) sprawia, że wszystkie informacje o danym obszarze sieci zostają wyczyszczone. Nie będzie możliwości ich odtworzenia. Pozostaje tylko sam obszar sieci.

Delete all and log out - wybór tej opcji (dostępne tylko dla super administratorów) sprawia, że wszystkie informacje o danym obszarze sieci zostaną usunięte – łącznie z samym obszarem i jego właścicielem. Po wykonaniu tych czynności nastąpi automatyczne

wylogowanie z systemu bez możliwości ponownego zalogowania do przed chwilą usuniętego obszaru.



Rysunek 4.15. Panel kont użytkowników

- **Help**

Index – opcja ta jak na razie nie jest dostępna. Być może powstanie w przyszłości. W trakcie działania programu użytkownik może wybrać przycisk *F1*, wtedy zostanie wyświetlona strona internetowa z raportem zawierającym informacje o strukturze bazy danych, jej tabelach oraz o diagramie przypadków użycia.

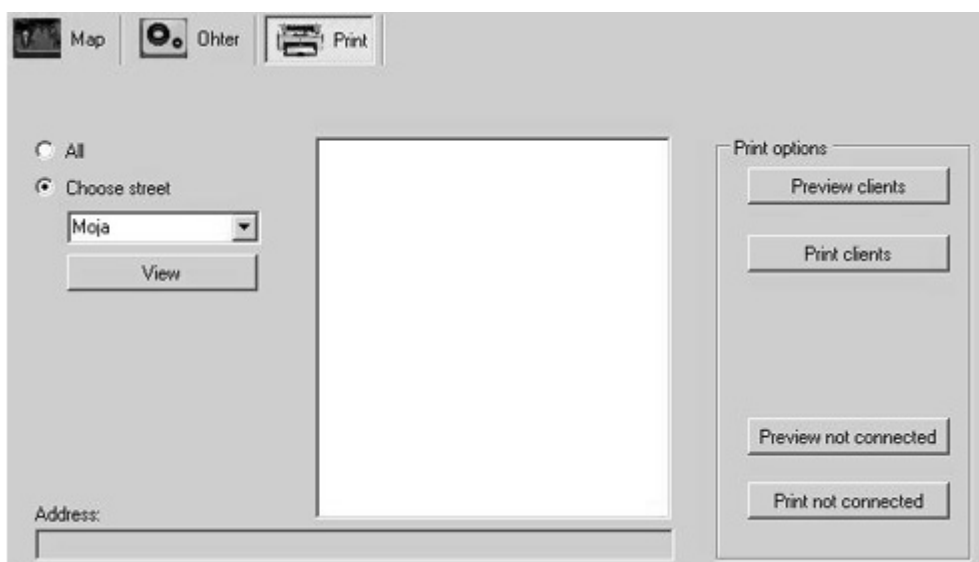
About - po wyborze tej opcji zostanie wyświetlone okno z podstawowymi informacjami na temat aplikacji *WebManagement* takich jak pełna nazwa programu, numer wersji, imię i nazwisko autora oraz adres e-mail autora.

W aplikacji *WebManagement* oprócz menu są dostępne również zakładki widoczne w głównym oknie aplikacji, które są opisane poniżej.

Map - po wyborze tej zakładki widoczny staje się panel z mapami, na których wszystko jest rysowane.

Other - zawiera zestaw paneli, które stają się widoczne po wyborze konkretnych funkcji programu. Po wybraniu tej zakładki początkowo żaden panel nie jest widoczny.

Print - prezentuje zestaw obiektów do komunikacji z użytkownikiem w celu dokonania wydruku (rysunek 4.16). Po lewej stronie rysunku widać możliwość wyboru wszystkich ulic, do których należą klienci lub jakiejś jednej konkretnej. Po wybraniu opcji *View* wyświetlona zostaje lista klientów. Po wyborze pojedynczego elementu z listy, zostaje w pasku *Address* wyświetlony pełny adres wybranego klienta. Przyciski *Preview Clients* oraz *Preview not connected* umożliwiają podgląd wydruku, a przyciski *Print clients* oraz *Print not connected* dają możliwość wydrukowania listy klientów w zależności od wybranych wcześniej opcji.



Rysunek 4.16. Dostępne zakładki dodatkowe (fragment okna)

5. Testowanie i uruchamianie

Testowanie to jeden z procesów kontroli jakości oprogramowania. W niniejszym rozdziale opisano, w jaki sposób testowany był system tworzony dla firmy *CzarNet* oraz jaki sprzęt został do tego celu wykorzystany. Zwrócono również uwagę na kilka problemów, które pojawiły się w czasie implementacji i opisano, w jaki sposób zostały rozwiązane. Na koniec zaprezentowano wykonanie przykładowego zadania za pomocą systemu powstałego w ramach niniejszej pracy.

5.1. Opis testowania

Testowanie systemu odbywało się systematycznie wraz z powstawaniem kolejnych funkcji oprogramowania. Każda funkcja była testowana początkowo przez autora. Testowanie polegało na wprowadzaniu danych o różnych wartościach, aby sprawdzić czy funkcje zostaną prawidłowo wykonane. Wielokrotnie testowano sprawdzanie poprawności danych tekstowych wprowadzanych przez użytkownika, aby do bazy dodawane były poprawne wartości tekstowe albo liczbowe. W czasie testowania wprowadzane były następujące dane:

- ciągi znaków (tekst),
- wartości liczbowe,
- puste znaki.

Dla każdego pola tekstowego lub innego obiektu, który służy do komunikacji z użytkownikiem następowała weryfikacja poprawności wpisywanych danych. Jeśli na przykład miała zostać wpisana liczba, a zamiast tego wpisano tekst albo nie wpisano żadnego znaku, wtedy testy pomagały znaleźć błędy.

Podczas testów postawionych było wiele punktów kontrolnych w programie (*breakpoints*), które pozwalały znaleźć błędy. To właśnie na etapie testowania kilka razy okazało się, że brakuje ważnych danych w bazie i że trzeba utworzyć jeszcze kilka tabel. Wiązało się to z ciągłym aktualizowaniem schematu bazy oraz źródła danych obiektu w programie. Ten pierwszy opisany powyżej etap testowania był stosowany z użyciem sprzętu, który opisano w poniższej tabeli.

| Numer | Procesor | Pamięć RAM | System operacyjny |
|-------|---------------------------------|------------|-------------------------|
| 1 | Intel Core 2 Duo 2.34 GHz | 2GB | Windows XP Professional |
| 2 | Intel Pentium Dual Core 2.0 GHz | 2GB | Windows XP Professional |
| 3 | Intel Celeron 1.5 GHz | 196MB | Windows XP Home on |

Oprócz tego typu testów odbywało się również sprawdzanie działania systemu w towarzystwie osoby, która będzie z niego korzystała. Ta faza sprawdzania polegała już nie na wykryciu błędów typowo programistycznych, ale raczej tych związanych z funkcjonalnością systemu. Podczas testowania okazało się, że potrzebnych byłoby jeszcze kilka dodatkowych modyfikacji. Na przykład zauważono, że przydałoby się aby w bazie danych był zapisywany adres MAC każdego urządzenia, dzięki czemu będzie wiadome, gdzie to urządzenie się znajduje. Będzie można łatwo je zlokalizować i sprawdzić czy jest ono własnością firmy *Czarnet* lub innej, która będzie korzystała z systemu.

5.2. Napotkane problemy

Poniżej opisano wybrane problemy, jakie pojawiły się podczas implementacji systemu dla firmy *CzarNet*, w jaki sposób je zaobserwowano, czego dotyczyły oraz przedstawiono sposób ich rozwiązywania.

Aktualizacja dostępnych danych podczas wyboru elementów

Problem powstał podczas próby zaimplementowania dodawania nowego połączenia pomiędzy sieciami. Podejmowane były różne próby, aby w trzech polach wyboru (określających sieci: źródłową, docelową oraz pośrednie) były możliwe do wyboru tylko takie dane (sieci), uzależnione między sobą (pomiędzy polami wyboru), które nie powodowałyby błędów i umożliwiały możliwą do wykonania operację. Początkowo autor próbował rozwiązać ten problem stosując wypełnianie każdego pola wyboru tą samą ilością sieci do wyboru – na przykład można było wybrać tą samą sieć jako źródłową i docelową. Jednak to znacznie utrudniało użytkowanie programu i powodowało błędy.

Problem został rozwiązany dzięki zastosowaniu kolekcji generycznych `BindingList<NazwaKlasy>`, które pozwalały uaktualniać dane i zachować brak sprzeczności pomiędzy poszczególnymi polami wyboru [7].

Dodawanie nowego bloku – rysowanie, zaznaczanie wierzchołków

Kolejny problem stanowiło dodawanie nowego bloku mieszkalnego a konkretnie zaznaczanie czterech wierzchołków tak, aby można go było narysować na mapie. Trudność polegała na tym, aby zaimplementować to w taki sposób, żeby umożliwić dodatkową funkcjonalność, a mianowicie rozpoznawanie obszarów bloków mieszkalnych (w czasie gdy bloki są wyświetlone na mapie), aby pokazać wszystkie informacje o danym bloku. Wybór wierzchołków nie jest rzeczą trywialną ze względu na późniejszą konieczność obliczania

układów równań. Na początku próbowano różnych sposobów, aby ograniczyć możliwość wyboru wierzchołków aż w końcu postanowiono rozwiązać problem stosując algorytm opisany w podrozdziale 3.5 tej pracy („Ustalanie wierzchołków bloku mieszkalnego”)

Wypełnianie kolorem obszaru bloku na mapie

Ten problem miał duży związek z opisanym w poprzednim podpunkcie. Początkowo (przy błędnym algorytmie) bloki mieszkalne nie były rysowane na mapie jako zamknięte figury czworokątne, lecz jako zupełnie inne powierzchnie, a to przez złe zaimplementowanie układów równań oraz złe dobieranie wierzchołków bloku. Zmiany wprowadzone w odpowiednim doborze wierzchołków oraz w rozwiązywaniu odpowiedniej nierówności pozwoliły rozwiązać ten problem.

Tworzenie połączenia między sieciami zgodnie z założeniami

Próbując zastosować się do założeń autor chciał stworzyć algorytm tworzenia połączenia pomiędzy sieciami, które uwzględniałoby automatyczne dobieranie ilości żył światłowodu. Jeśli wolnych żył światłowodu by już nie było, to połączenie byłoby niemożliwe do zrealizowania. Początkowo raczej bez większych problemów udało się zaimplementować tę funkcję, ale tylko w przypadku, gdy połączenie jest tworzone od jednej sieci do drugiej bez rozgałęzień (topologia magistrali). Natomiast problem zaczął się przy rozgałęzieniach (topologia gwiazdy i gwiazdy rozszerzonej). Problemy stwarzało przede wszystkim sprawdzanie, czy istnieją jeszcze wolne światłowody. Istniała w związku z tym potrzeba sprawdzania wszystkich końcowych światłowodów w sieciach. Następnie trzeba było poprawnie przyporządkować ilość żył w światłowodach dla aktualnie dodawanych połączeń.

W tym miejscu pojawiało się mnóstwo błędów, złych obliczeń. Opis algorytmu, który udało się opracować znajduje się w podrozdziale 3.5 („Tworzenie połączenia sieciowego”).

Anulowanie rysowania ulicy i bloku

Początkowo program był tak skonstruowany, że podczas dodawania ulicy oraz bloku mieszkalnego nie było możliwe anulowanie wykonywania operacji rysowania. Trzeba ją było dokończyć – w przeciwnym wypadku zgłaszany był wyjątek systemowy. Jednak istniała potrzeba, aby dodać tę ważną funkcję. Nie było to łatwe. Powstawały problemy dotyczące między innymi odświeżania rysowania fragmentów ulicy. Aplikacja zgłaszała wyjątek systemowy podczas wielokrotnego naciskania przycisku *Delete* w celu anulowania

rysowania, ponieważ operacje usuwania z bazy nie były dobrze zaimplementowane. Te błędy udało się dość sprawnie poprawić dzięki badaniu kodu krok po kroku oraz dzięki poprawieniu procedur usuwania informacji o wierzchołkach ulic i samych ulic. Analogiczne problemy związane były z dodawaniem bloków mieszkalnych. Zostały usunięte w podobny sposób.

Reagowanie na przyciski klawiatury w trakcie działania programu

Po zaimplementowaniu funkcji, która odpowiadała za reagowanie na wybieranie przycisków klawiatury (chodzi tu o *Backspace*, *Delete*, *Right*, *Left*) okazało się, że po uruchomieniu aplikacji nie było możliwe wpisanie tekstu w żadne pole tekstowe ani inne miejsce do tego przewidziane.

Aby rozwiązać ten problem autor dokładnie przeanalizował kod programu, sprawdził właściwości obiektów klasy `TextBox`. Po głębszej analizie okazało się, że jedna z funkcji zwraca niepoprawną wartość. Jest to funkcja `protected override Boolean ProcessCmdKey(ref Message msg, Keys keyData)`.

Początkowo zwracała ona wynik `true`, ale okazało się, że wtedy cały czas sprawdza ona co zostaje wybrane na klawiaturze i ‘pochłania’ wszystkie znaki. Zmiana zwracanego wyniku na `false` pozwoliła uporać się z problemem.

Puste kolekcje

Dość częstym problemem było to, że gdy kolekcja elementów pobranych z bazy nie zawierała żadnych elementów, to w miejscu, w którym te elementy miały zostać wyświetlone albo miały być na nich przeprowadzane operacje, zgłaszany był wyjątek systemowy i działanie aplikacji było przerywane. Było tak podczas korzystanie z *LINQ*, który pozwalał zbierać do kolekcji rekordy z bazy. Problem został rozwiązany we wszystkich przypadkach dzięki zastosowaniu pętli:

```
foreach (var zbiór in kolekcja)
{
    zbiór operacji na danych;
}
```

Dla przykładowej kolekcji:

```
var kolekcja =
    from blok in this.siecDataSet.Nlok
    where warunek
    select new { blok };
```

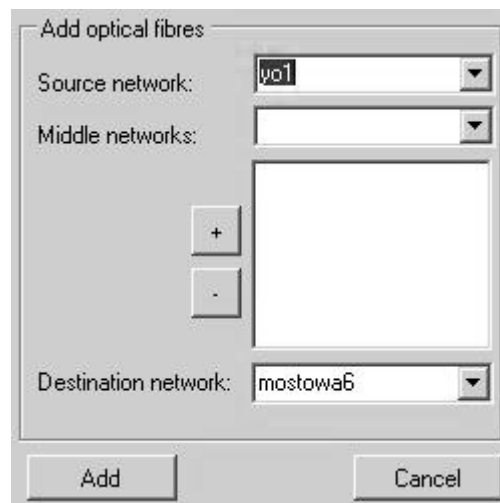
Dzięki takiemu rozwiązaniu przy ilości elementów w zbiorze równym 0 na danych nie były wykonywane żadne operacje, co z kolei nie powodowało dalszego zgłaszania wyjątków systemowych.

5.3. Rozwiązanie przykładowego zadania za pomocą programu

Poniżej zostanie przedstawione rozwiązanie przykładowego zadania, jakim jest dodawanie nowego połączenia pomiędzy sieciami

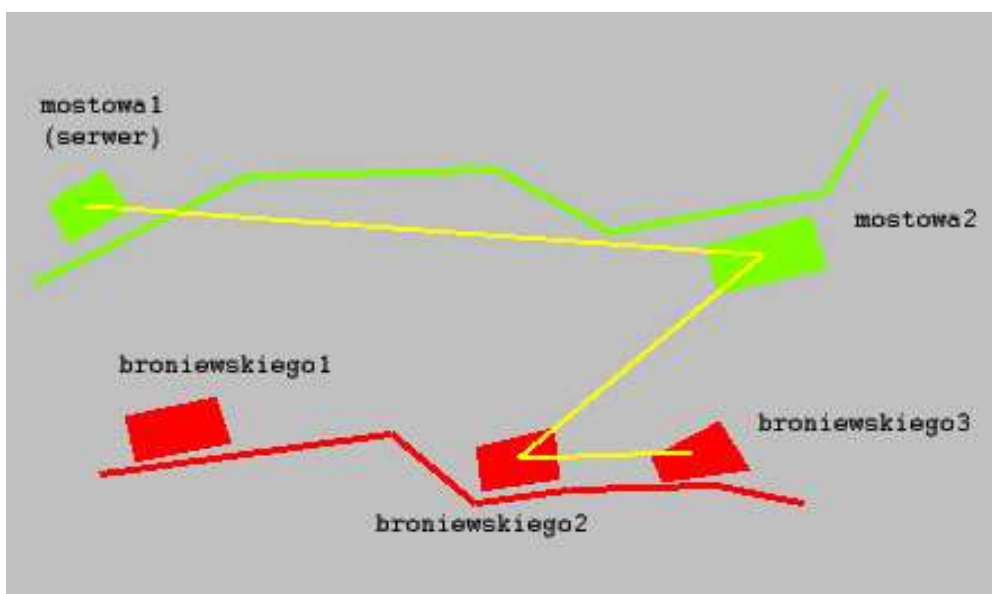
Autor zakłada, że wcześniej został stworzony obszar sieci o nazwie „Leszczyny Osiedle”, który ma przypisanego swojego właściciela. Ten właściciel utworzył już na mapie nowe ulice o nazwach: „Mostowa” oraz „Broniewskiego”. Do każdej ulicy dodał bloki, a w nich sieci komputerowe. Zakłada się dodatkowo, że na Mostowej są 2 bloki mieszkalne (numer 1, gdzie nazwa sieci to „mostowa1” oraz numer 2, gdzie nazwa sieci to „mostowa2”) natomiast na ulicy Broniewskiego są 3 bloki mieszkalne (numer 1, gdzie nazwa sieci to „broniewskiego1”; numer 2, gdzie nazwa sieci to „broniewskiego2”; numer 3, gdzie nazwa sieci to „broniewskiego3”). Serwerownia została założona w sieci „mostowa1”. Ostatnie założenie jest takie, iż żadne sieci nie są jeszcze połączone kablem. Pozostałe szczegóły nie są istotne w celu demonstracji tego zadania.

Użytkownik chcąc stworzyć nowe połączenie wybiera w menu głównym programu zakładkę *Add* i wybiera opcję *Connection*. Następnie zostaje wyświetlony panel widoczny na rysunku 5.2, gdzie wybierana jest sieć źródłowa, docelowa oraz sieci pośrednie. Z listy *Source network* użytkownik wybiera sieć źródłową. Ponieważ żadna sieć nie ma jeszcze połączenia, więc jedyną możliwą siecią do wyboru będzie ta, w której znajduje się serwerownia, czyli „mostowa1”. Gdy użytkownik ją wybierze, wtedy uzupełni się zawartość listy *Destination network*. Na tej liście zostają wyświetlone do wyboru tylko te sieci, które nie mają jeszcze połączenia z innymi. Użytkownik wybiera jedną z pozostałych na przykład „broniewskiego3”. Po dokonaniu tego wyboru uzupełni się zawartość listy wyboru *Middle networks*, z której będzie można wybrać sieci pośrednie, przez które będzie przebiegał światłowód, a które również nie mają jeszcze połączenia sieciowego.



Rysunek 5.2. Przykład tworzenie nowego połączenia między sieciami

Gdy użytkownik zdecyduje się na wybór jako sieci pośrednich „mostowa2” i „broniewskiego2”, wtedy zaznacza je w polu wyboru i wybiera opcję „+” – zostają one wtedy dodane do listy widocznej w środkowej części panelu. Można je również anulować wybierając „-”. Jeżeli użytkownik wybierze opcję *Add* wtedy połączenie zostanie dodane, o ile jest możliwe do zrealizowania pod względem ilości żył w światłowodzie. Załóżmy, że połączenie dodano. Po wyborze opcji wyświetlania można zobaczyć bloki oraz logiczne połączenia światłowodami pomiędzy sieciami, jak zostało to zaprezentowane na rysunku 5.3.



Rysunek 5.3. Realizacja przykładu w programie

Podsumowanie

Celem niniejszej pracy inżynierskiej było stworzenie systemu wspomagającego zarządzanie informacją o sieci internetowej, realizowanego na potrzeby firmy *CzarNet*. Opracowany system pozwala na inwentaryzację obiektów sieciowych oraz obiektów administracyjnych razem z ich parametrami. Dzięki niemu można także określić geograficzne położenie obiektów administracyjnych (ulica, blok mieszkalny, klatka w bloku mieszkaldnym) oraz udostępnić informacje o tym, które budynki należą do danej podsieci, jacy klienci z nich korzystają czy też jak wyglądają logiczne połączenia światłowodami pomiędzy sieciami.

System wspomagający zarządzanie informacją o sieci internetowej, tworzony w ramach niniejszej pracy to narzędzie, które może okazać się przydatne w niejednej firmie. Ciagle rośnie liczba firm, które oferują podłączenia do sieci klientom, którzy będą płacić okresowo abonament w stałej wysokości i uzyskają w zamian nielimitowany czasem dostęp do Internetu. System został wdrożony w firmie *CzarNet*, z siedzibą w Leszczynach. Służy on jako oprogramowanie dodatkowe, ponieważ oprócz bazy klientów potrzebne jest narzędzie, dzięki któremu można w szybki i łatwy sposób zorientować się, gdzie znajdują się różne elementy sieci oraz elementy administracyjne. Przykładem uzmysławiającym praktyczne wykorzystanie oprogramowania pomocniczego może być możliwość szybkiej weryfikacji różnych danych klienta. Za pomocą systemu można sprawdzić, gdzie na mapie znajduje się blok, w którym mieszka klient oraz ustalić przy jakiej ulicy on się znajduje.

Niewątpliwą zaletą powstałego oprogramowania jest także ograniczenie możliwości kupowania przez firmę skradzionych urządzeń takich jak switch czy router. Zdarzało się, że wskutek niedopatrzeń firma *CzarNet* kupowała używane urządzenia, po czym okazywało się, że są one jej własnością, jednak wcześniej zostały skradzione. Obecnie w bazie danych zapisany jest każdy adres MAC urządzenia oraz wiadomo, gdzie ono się znajduje, dzięki czemu można łatwo je zlokalizować i sprawdzić czyją są własnością.

Jako potencjalne kierunki rozwoju opracowanego systemu można wymienić połączenie tego oprogramowania, służącego na razie jako dodatkowe z oprogramowaniem głównym, czyli tym, które operuje na głównej bazie klientów. Pozwoliłoby to zwiększyć możliwości zarządzania firmą. Gdy to się uda, na pewno zaistnieje potrzeba przyspieszenia wyszukiwania elementów w bazie danych, ponieważ będzie ich znacznie więcej. W najbliższej przyszłości planowane jest wykorzystanie do tego celu projektu typu *Open Source* o nazwie *Lucene.NET*, który pozwala indeksować dane z bazy na dysku lokalnym i dzięki temu przyspieszyć ich wyszukiwanie.

System jest napisany w taki sposób, że mogą z niego korzystać administratorzy sieci i osoby przez nich wyznaczone. Wszystkie dane są przechowywane w bazie danych, która znajduje się na serwerze Microsoft SQL Server. Z tym serwerem łączy się aplikacja, z której korzysta użytkownik. Nie jest to oprogramowanie przeznaczone dla klientów. Jednak każdy użytkownik potrzebowałby pomocy w korzystaniu z niektórych funkcji programu, dlatego autor ma nadzieję, że uda się dodać w przyszłości kolejny ważny element – pomoc kontekstową, która z pewnością ułatwi pracę administratorom korzystającym z systemu wspomagającego zarządzanie informacją o sieci internetowej.

Powstały system nie jest w stanie konkurować z systemami paszportyzacji, które posiadają ogromne firmy telekomunikacyjne, ale mniejsze firmy mogłyby zdaniem autora odnieść z niego korzyść. Autor pracy ma nadzieję, że w przyszłości również inne firmy będą chciały użytkować napisane przez niego oprogramowanie.

Literatura

1. Avery J., *100 sposobów na Visual Studio*, Helion, Gliwice, 2005.
2. Liberty J., *Programowanie C#*, Helion, Gliwice, 2006.
3. Perry S., *C# i .NET*, Helion, Gliwice, 2006.
4. Waymire R., Sawtell R., *SQL Server 2000 dla każdego*, Helion, Gliwice, 2002.
5. *Architektura ADO.NET*, http://download.microsoft.com/download/2/3/f/23f09d42-8d97-4229-9f7c-6c600598c027/PDD2004_Konflikty.ppt (dostępny - luty 2009).
6. *Kabel światłowodowy – opis*,
<http://www.idg.pl/slownik/termin/33284/kabel.swiatlowodowy.html> (dostępny - styczeń 2009).
7. *Kolekcje generyczne w C#*, <http://msdn.microsoft.com/en-us/library/ms132679.aspx> (dostępny - styczeń 2009).
8. *Kurs języka SQL*, http://www.centrumxp.pl/dotNet/21,1,kategoria,Kurs_SQL.aspx (dostępny - listopad 2008).
9. *Kurs UML*, http://serwis.magazynyinternetowe.pl/artukul/3542,1,1242,kurs_uml_-_czesc_1_-_wstep_i_diagramy_klas.html (dostępny - listopad 2008).
10. *Opis LINQ*, www.si.pjwstk.edu.pl/dydaktyka/mgr/2005-2006-winter/The_LINQ_Project.ppt (dostępny - luty 2009).
11. *Topologia gwiazdy – opis*, <http://apgar.w.interia.pl/tgwiazda.htm> (dostępny - styczeń 2009).
12. *Topologia gwiazdy rozszerzonej – opis*,
http://members.lycos.co.uk/spookypl/html/topologie_fizyczne.htm (dostępny - styczeń 2009).
13. *Zalety języka C#*, <http://www.pcworld.pl/artykuly/48591/Cztery.plusy.html> (dostępny - styczeń 2009).

Dodatek A. Fragment skryptu generującego bazę danych

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[Dostep]') AND
type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Dostep](
    [IDC] [int] IDENTITY(1,1) NOT NULL,
    [IDM] [int] NOT NULL,
    [IDU] [int] NOT NULL,
    CONSTRAINT [PK_Dostep] PRIMARY KEY CLUSTERED
(
    [IDC] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[EdytujMiejscowosc]') AND type in (N'P', N'PC'))
BEGIN
EXEC dbo.sp_executesql @statement = N'

CREATE PROCEDURE [dbo].[EdytujMiejscowosc]
    @IDM int out,
    @LokMap varchar(100),
    @LokEarth varchar(100)
AS
UPDATE Miejscowosc
SET LokMap = @LokMap, LokEarth = @LokEarth
WHERE (@IDM = IDM)

'
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[Urzadzenie]')
AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Urzadzenie](
    [IDurz] [int] IDENTITY(1,1) NOT NULL,
    [Rodzaj] [varchar](50) NOT NULL,
    CONSTRAINT [PK_Urzadzenie] PRIMARY KEY CLUSTERED
(
    [IDurz] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[EdytujSiec]')
AND type in (N'P', N'PC'))
BEGIN
EXEC dbo.sp_executesql @statement = N'
```

Dodatek B. Spis tabel bazy danych

Dostęp(**IDC**, **IDU**, **IDM**)

IDC (Typ: int, Rola: Primary Key, Opis: Identyfikator dostępu, inkrementowany automatycznie).

IDU (Typ: int, Opis: Identyfikator określający użytkownika)

IDM(Typ: int, Opis: Identyfikator określający miejscowość, do której ma dostęp dany użytkownik)

Miejscowość(**IDM**, **Nazwa**, **LokEarth**, **LokMap**, **Miasto**, **Kod**)

IDM (Typ: int, Rola: Primary Key, Opis: Identyfikator miejscowości, inkrementowany automatycznie).

LokEarth (Typ: varchar, Opis: Określa fizyczne położenie pliku z widokiem satelitarnym obszaru na dysku).

LokMap (Typ: varchar, Opis: Określa fizyczne położenie pliku z mapą obszaru na dysku).

Miasto (Typ: varchar, Opis: Określa nazwę miasta, w którym tworzony jest obszar sieci)

Kod (Typ: varchar, Opis: Kod pocztowy miejscowości, w której tworzony jest obszar sieci).

Ulica(**IDU**, **IDM**, **Nazwa**, **Komentarz**, **Kolor**)

IDU (Typ: int, Rola: Primary Key, Opis: Identyfikator ulicy, inkrementowany automatycznie).

IDM (Typ: int, Rola: Foreign Key, Opis: Określa do jakiej miejscowości należy dana ulica).

Nazwa (Typ: varchar, Opis: Nazwa ulicy).

Komentarz (Typ: varchar, Opis: Komentarz, może mieć wartość pustą (null)).

Kolor (Typ: varchar, Opis: Nazwa koloru, który jest przypisany do ulicy, wybrany przez użytkownika).

UlicaXY(**IDUxy**, **IDU**, **X**, **Y**)

IDUxy (Typ: int, Rola: Primary Key, Opis: Identyfikator współrzędnych ulicy, inkrementowany automatycznie).

IDU (Typ: int, Rola: Foreign Key, Opis: (Określa do jakiej ulicy należy współrzędna).

X (Typ: int, Opis: Pierwsza współrzędna położenia wierzchołka na mapie).

Y (Typ: int, Opis: Druga współrzędna położenia wierzchołka na mapie).

Blok(**IDB**, **IDU**, **Numer**, **Komentarz**)

IDB (Typ: int, Rola: Primary Key, Opis: Identyfikator bloku, inkrementowany automatycznie).

IDU (Typ: int, Rola: Foreign Key, Opis: Określa do jakiej ulicy należy dany blok).

Numer (Typ: varchar, Opis: Numer bloku).

Komentarz (Typ: varchar, Opis: Komentarz do bloku, może mieć wartość pustą).

BlokXY(**IDBxy**, **IDB**, **x1**, **y1**, **x2**, **y2**, **x3**, **y3**, **x4**, **y4**)

IDBxy (Typ: int, Rola: Primary Key, Opis: Identyfikator wierzchołków bloku, inkrementowany automatycznie).

IDB (Typ: int, Rola: Foreign Key, Opis: Określa blok, do którego należą dane współrzędne).

x1, **x2**, **x3**, **x4** (Typ: int, Opis: Współrzędne X na każdy z czterech wierzchołków).

y1, **y2**, **y3**, **y4** (Typ: int, Opis: Współrzędne Y na każdy z czterech wierzchołków)

Siec(**IDS**, **IDTypUrz**, **IDB**, **Nazwa**, **AdresIP**, **Komentarz**)

IDS (Typ: int, Rola: Primary Key, Opis: Identyfikator sieci, inkrementowany automatycznie).

IDTypUrz (Typ: int, Rola: Foreign Key, Opis: Określa jaki typ urządzenia jest używany w danej sieci).

IDB (Typ: int, Rola: Foreign Key, Opis: Określa do jakiego bloku należy dana sieć).

Nazwa (Typ: varchar, Opis: Nazwa sieci).

AdresIP (Typ: varchar, Opis: Adres IP sieci).

Komentarz (Typ: varchar, Opis: Komentarz do sieci, może mieć wartość pustą (null)).

Światłowód(**IDsw**, **IDS**, **j**, **IDnext**, **Pop**, **ostatni**)

IDsw (Typ: int, Rola: Primary Key, Opis: Identyfikator światłowodu, inkrementowany automatycznie).

IDS (Typ: int, Rola: Foreign Key, Opis: Określa do jakiej sieci jest połączony światłowód).

j (Typ: int, Opis: Ilość żył w światłowodzie w aktualnej sieci).

IDnext (Typ: int, Opis: Identyfikator określający sieć, do której prowadzi dany światłowód).

Pop (Typ: int, Opis: Identyfikator określający sieć źródłową, od której jest poprowadzony dany światłowód. Przyjmuje wartość 0 jeśli nie ma poprzednika).

Ostatni (Typ: int, Opis: Określa czy sieć jest wewnątrz połączenia pomiędzy innymi sieciami (wtedy przyjmuje wartość 0) czy może jest ostatnią siecią do której dociera dany światłowód (wtedy przyjmuje wartość 1). Wartości 0 i 1 to jedyne, które są możliwe do wyboru).

Urządzenie(IDurz, Rodzaj)

IDurz (Typ: int, Rola: Primary Key, Opis: Identyfikator urządzenia, inkrementowany automatycznie).

Rodzaj (Typ: varchar, Opis: Nazwa rodzaju urządzenia).

UrządzenieTyp(IDtyp, IDurz, Typ)

IDtyp (Typ: int, Rola: Primary Key, Opis: Identyfikator typu urządzenia, inkrementowany automatycznie).

IDurz (Typ: int, Rola: Foreign Key, Opis: Określa jakiego rodzaju urządzeniem jest konkretny model urządzenia).

Typ (Typ: varchar, Opis: Nazwa konkretnego modelu urządzenia określonego rodzaju).

Klatka(IDK, IDB, Nazwa, Komentarz)

IDK (Typ: int, Rola: Primary Key, Opis: Identyfikator klatki, inkrementowany automatycznie).

IDB (Typ: int, Rola: Foreign Key, Opis: Określa do jakiego bloku mieszkalnego należy klatka).

Nazwa (Typ: varchar, Opis: Nazwa klatki).

Komentarz (Typ: varchar, Opis: Komentarz do klatki, może przyjmować wartość pustą).

Osoba(IDO, Imie, Nazwisko)

IDO (Typ: int, Rola: Primary Key, Opis: Identyfikator osoby, inkrementowany automatycznie).

Imie (Typ: varchar, Opis: Określa imię osoby).

Nazwisko (Typ: varchar, Opis: Określa nazwisko osoby).

Uzytkownik(IDU, IDO, Login, Haslo, Rola)

IDU (Typ: int, Rola: Primary Key, Opis: Identyfikator użytkownika, inkrementowany automatycznie).

IDO (Typ: int, Rola: Foreign Key, Opis: Określa przyporządkowanie osoby do użytkownika).

Login (Typ: varchar, Opis: Nazwa użytkownika w systemie).

Haslo (Typ: varchar, Opis: Hasło użytkownika w systemie).

Rola (Typ: int, Opis: Określa poziom uprawnień użytkowników w systemie. Może przyjmować wartości 0, 1 lub 2.

0 – administrator

1 – zwykły użytkownik

2 – super administrator

Klient(IDklient, IDO, IDklatki, IP, NrMieszkania)

IDklient (Typ: int, Rola: Primary Key, Opis: Identyfikator klienta, inkrementowany automatycznie).

IDO (Typ: int, Rola: Foreign Key, Opis: Określa przyporządkowanie osoby do klienta).

IDklatki (Typ: int, Rola: Foreign Key, Opis: Określa w jakiej klatce mieszka dany klient).

IP (Typ: varchar, Opis: Adres IP, który posiada klient).

NrMieszkania (Typ: int, Opis: Numer mieszkania danego klienta).

Mac(IDmac, Adres, IDsiec, IDtyp)

IDmac (Typ: int, Rola: Primary Key, Opis: Identyfikator adresu MAC, inkrementowany automatycznie).

Adres (Typ: varchar(12), Opis: Określa fizyczny adres urządzenia).

IDSiec (Typ: int, Opis: Określa sieć do jakiej należy urządzenie).

IDtyp (Typ: int, Opis: Określa identyfikator urządzenia).

Dodatek C. Procedury SQL

| Nazwa procedury | Opis | Parametry |
|-------------------------|--|--|
| ClearBlokDown | Usuwanie bloku i wszystkich dziedziczących po nim elementów – współrzędnych określających położenie oraz klatek mieszkalnych. | IDB – Identyfikator bloku (wejście) |
| ClearClient | Usuwanie klienta należącego do klatki, której identyfikator przekazywany jest w parametrze. | IDK – Identyfikator klatki (wejście) |
| ClearSiec | Usuwanie wszystkich informacji związanych z siecią o identyfikatorze podanym w parametrze. Usuwana jest sieć oraz światłowody do niej należące | IDS – Identyfikator sieci |
| ClearUlica | Usuwanie ulicy o podanym identyfikatorze wraz z jej współrzędnymi | IDU – Identyfikator ulicy (wejście) |
| DodajBlokXY | Dodawanie współrzędnych do nowo powstałego bloku. | IDBxy – identyfikator wierzchołków (wyjście) IDB – identyfikator bloku (wejście, wyjście) x1, x2, x3, x4, y1, y2, y3, y4 – współrzędne (wejścia) |
| DodajKlient | Dodawanie klienta do już istniejącego bloku, istniejącej klatki. | IDklient – identyfikator klienta (wyjście) IDO – identyfikator osoby (wejście, wyjście) IDklatki – identyfikator klatki (wejście, wyjście) IP, NrMieszkania, Imie, Nazwisko – (wejścia) |
| DodajKlatka | Dodawanie klatki do istniejącego bloku | IDK – identyfikator klatki (wyjście) IDB – identyfikator bloku (wejście, wyjście) Nazwa, Komentarz (wejścia) |
| DodajMsc | Dodawanie nowej miejscowości, w której może istnieć wiele obszarów sieci | IDM – identyfikator miejscowości (wyjście) Nazwa, Miasto, Kod – (wejścia) |
| DodajUlica | Dodawanie nowej ulicy do istniejącej miejscowości. | IDU – identyfikator IDM – identyfikator miejscowości, do której należy ulica Nazwa, Komentarz, Kolor – (wejścia) |
| DodajUlicaXY | Dodawanie punktu określającego położenie wierzchołka ulicy na mapie terenu | IDUxy – identyfikator wierzchołków ulicy (wyjście) IDU – identyfikator ulicy, do której będą przypisywane wierzchołki X, Y – współrzędne pojedynczego wierzchołka ulicy |
| DodajUzytkownika | Dodawanie nowego użytkownika połączone z dodawaniem nowej | IDU – identyfikator użytkownika |

| | | |
|--------------------------|---|--|
| | osoby do systemu | (wyjście) IDO – identyfikator osoby, która staje się użytkownikiem systemu (wejście, wyjście) Login, Hasło, Rola – (wejścia) |
| EdytujSwiatlowod2 | Edytowanie sieci polegająca na zmianie ilości żył w światłowodzie dochodzącym do danej sieci | IDS – identyfikator sieci, określający sieć, która będzie edytowana (wejście) j – określenie nowej ilości żył światłowodu |
| EdytujBlok | Zastąpienie starego numeru bloku nowym numerem | IDB – identyfikator bloku, który będzie podlegał edycji (wejście) Numer – nowy numer bloku (wejście) |
| EdytujKlatka | Zastąpienie starej nazwy klatki nową nazwą. | IDK – identyfikator klatki, która będzie podlegała edycji (wejście) Nazwa – nowa nazwa dla klatki (wejście) |
| EdytujMiejscowosc | Zastąpienie starych ścieżek do plików graficznych na dysku (mapy terenu, obszaru sieci) nowymi ścieżkami. | IDM – identyfikator miejscowości, która będzie edytowana (wejście) LokMap, LokEarth – nowe nazwy ścieżek do plików graficznych z mapami (wejścia) |
| EdytujOsoba | Zastąpienie poprzedniego imienia i nazwiska przyporządkowanego do danej osoby nowymi wartościami. | IDO – identyfikator osoby, która będzie podlegać edycji Imie, Nazwisko – nowe ciągi znaków określające imię oraz nazwisko osoby (wejścia) |
| EdytujOstatni | Zamiana wartości oznaczającej to, czy dany światłowód jest światłowodem ostatnim, kończącym | Pop – identyfikator określający poprzednika czyli sieć źródłową dla danego światłowodu (wejście) Ostatni – nowa wartość 0 lub 1 dla określenia czy dany światłowód jest ostatnim, kończącym (wejście) |
| EdytujSiec | Edytowanie komentarza do sieci | IDS – identyfikator sieci, która będzie edytowana (wejście) Komentarz – nowy komentarz (wejście) |
| UBlok | Usuwanie bloku mieszkalnego oraz wszystkich informacji o położeniu jego wierzchołków. | IDB – identyfikator bloku, który ma zostać usunięty (wejście) |
| UsunBlok | Usuwanie bloku mieszkalnego bez usuwania informacji o położeniu jego wierzchołków | IDB – identyfikator bloku, który ma zostać usunięty (wejście) |
| UsunBlokXY | Usuwanie wierzchołków określających położenie bloku mieszkalnego na mapie (bez usuwania samego bloku) | IDB – identyfikator bloku, którego wierzchołki mają zostać usunięte (wejście) |
| UsunKlienta | Usunięcie klienta z bazy danych | IDklient – identyfikator klienta, który ma zostać usunięty (wejście) |
| UsunKlatka | Usunięcie klatki w bloku mieszkalnym | IDK – identyfikator klatki, która ma zostać usunięta (wejście) |
| UsunNetwork | Usunięcie miejscowości (obszaru sieci) o podanym identyfikatorze | IDM – identyfikator miejscowości (która określa obszar sieci – wejście) |
| UsunSiec | Usunięcie danej sieci | IDS – identyfikator sieci |

| | | |
|--------------------------|--|---|
| | | przeznaczonej do usunięcia (wejście) |
| UsunTyp | Usuwanie typu urządzenia o podanym identyfikatorze | IDtyp – identyfikator typu urządzenia, które ma zostać usunięte (wejście) |
| UsunTypUrzadzenia | Usuwanie typu urządzenia o podanym identyfikatorze | IDtyp – identyfikator typu urządzenia, które ma zostać usunięte (wejście, wyjście) |
| UsunUlicaXY | Usuwanie pojedynczego wierzchołka, wchodzącego w skład określenia położenia ulicy na mapie terenu. | IDUxy – identyfikator współrzędnej ulicy, która ma zostać usunięta (wejście) |
| UsunUrzadzenie | Usuwanie konkretnego rodzaju urządzenia wraz ze wszystkimi modelami które do niego należą. | IDurz – identyfikator urządzenia, które ma zostać usunięte (wejście) |
| UsunDostep | Usuwanie z systemu informacji o dostępie użytkownika do bazy w danym obszarze sieci. | IDU – identyfikator użytkownika, którego dostęp ma zostać zabrany (wejście) |
| UsunUser | Usuwanie z systemu użytkownika o podanym identyfikatorze osoby | IDO – identyfikator osoby (wejście) |
| UsunUzytkownik | Usuwanie z systemu użytkownika | IDU – identyfikator użytkownika, który ma zostać usunięty (wejście) |

Dodatek D. Fragmenty kodu źródłowego

D1. Ustalanie wierzchołków bloku mieszkalnego

```
public void newBlock(object sender, MouseEventArgs e)
{
    //zaznaczanie punktów pomocniczych
    Graphics points = mainW.mapaPanel.CreateGraphics();
    float grubosc = 3f;
    Color c = Color.Black;
    Pen pen = new Pen(c, grubosc);
    points.DrawLine(pen, e.X, e.Y, e.X+1, e.Y+1);
    //ustalenie pierwszego wierzchołka
    if (counter < 3)
    {
        tabX[counter] = (int)(e.X);
        tabY[counter] = (int)(e.Y);
        if (counter == 0)
        {
            counter++;
        }
        //sprawdzanie warunku dla drugiego wybranego wierzchołka
        else if (counter == 1)
        {
            if ((tabX[1] > tabX[0]) && (tabY[1] > tabY[0]))
            {
                counter++;
            }
            else
            {
                mainW.myStatusStripLabel1.Text = "";
                mainW.myStatusStripLabel1.Text = "Bad location";
            }
        }
        //sprawdzanie warunku dla trzeciego wybranego wierzchołka
        else if (counter == 2)
        {
            if ((tabX[1] > tabX[2]) && (tabY[2] > tabY[1]))
            {
                counter++;
            }
            else
            {
                mainW.myStatusStripLabel1.Text = "";
                mainW.myStatusStripLabel1.Text = "Bad location";
            }
        }
    }
}
//ustalenie ostatniego wierzchołka
else
{
    tabX[counter] = (int)(e.X);
    tabY[counter] = (int)(e.Y);

    if ((tabX[3] < tabX[0]) && (tabX[3] < tabX[2]) && (tabY[3] < tabY[2]) && (tabY[3] >
    tabY[0]))
    {
        int idbxy = 1;
        int idb = MainWindow.idb;
        siecDataSetTableAdapters.QueriesTableAdapter proc = new
            WebManagement.siecDataSetTableAdapters.QueriesTableAdapter();

        proc.DodajBlokXY(ref idbxy, ref idb, tabX[0], tabY[0], tabX[1], tabY[1],
            tabX[2], tabY[2], tabX[3], tabY[3]);

        siecDataSetTableAdapters.BlokTableAdapter blokTableAdapter = new
            WebManagement.siecDataSetTableAdapters.BlokTableAdapter();
        blokTableAdapter.Fill(mainW.siecDataSet.Blok);
        siecDataSetTableAdapters.BlokXYTableAdapter blokxyTableAdapter = new
            WebManagement.siecDataSetTableAdapters.BlokXYTableAdapter();
        blokxyTableAdapter.Fill(mainW.siecDataSet.BlokXY);
    }
}
```



```

        endEvent();
    }
    else
    {
        flag = 1;
        MessageBox.Show("Try again");
        endEvent();
    }
}
}
}

```

D2. Tworzenie bazy danych

```

private void createButton_Click(object sender, EventArgs e)
{
    List<int> lista;
    lista = new List<int>();

    CheckFormat checkFormat;
    checkFormat = new CheckFormat();
    lista = checkFormat.checkDatabaseFormat(databaseNameTextBox.Text,
                                            serverTextBox.Text);

    if (lista.Count != 0)    //jeśli są błędy
    {
        MessageBox.Show("There are some format errors");
        return;
    }
    else
    {
        string conStr;
        conStr = "Data Source=" + serverTextBox.Text + ";Initial Catalog=tempdb" +
                ";Integrated Security=True";
        SqlConnection conn = new SqlConnection(conStr);
        try
        {
            conn.Open();
        }
        catch (SqlException ex)
        {
            MessageBox.Show(ex.Message);
            return;
        }
        SqlCommand cmd2 = conn.CreateCommand();

        cmd2.Connection = conn;
        cmd2.CommandText = "EXEC sp_detach_db @dbname = 'siec'" +
            "EXEC sp_attach_single_file_db @dbname = 'yol', @physname = '" +
            openFileDialog1.FileName + "'";

        cmd2.CommandType = CommandType.Text;
        try
        {
            cmd2.ExecuteNonQuery();
        }
        catch (SqlException ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}
}

```

D3. Sprawdzanie czy istnieją wolne światłowody

```

/// <summary>
/// Stała oznaczająca ilość żył światłowodu odchodzących na jedną sieć
/// </summary>
public const int Cable = 2;

```

```

/// <summary>
/// Definicja klasy do przechowywania informacji o wolnych krańcach gdzie /// można podpiąć
sieć
/// </summary>
public class Ending
{
    public int IDSieciG;
    public int popG;
    public int jG;

    public int IDSieci
    {
        get { return IDSieciG; }
    }
    public int pop
    {
        get { return popG; }
    }
    public int j
    {
        get { return jG; }
    }
    /// <summary>
    /// Konstruktor klasy Ending
    /// </summary>
    /// <param name="ids"></param>
    /// <param name="p"></param>
    /// <param name="zyly"></param>
    public Ending(int ids, int p, int zyly)
    {
        IDSieciG = ids;
        popG = p;
        jG = zyly;
    }
}
public Ending k;

private int conn(int source, int dlugosc)
{
    int dl = dlugosc;
    int flaga = 0;
    int licz = 0;
    int pop = 0;
    int x;

    var polaczenia =
        from u in mainW.siecDataSet.Ulica
        join b in mainW.siecDataSet.Blok
        on u.IDU equals b.IDU
        where u.IDM == Login.IDmiejscowosc
        join s in mainW.siecDataSet.Siec
        on b.IDB equals s.IDB
        join siec in mainW.siecDataSet.Swiatlowod
        on s.IDS equals siec.IDS
        where siec.ostatni == 1
        select new { siec };

    if (polaczenia.Count() == 0)
    {
    }
    foreach (var po in polaczenia)
    {
        flaga = 0;
        licz = po.siec.IDnext; //było IDS
        pop = po.siec.Pop;
        if ((pop != 0) && (pop != po.siec.IDS))
        {
            while (flaga == 0)
            {
                var szukaj =
                    from sw in mainW.siecDataSet.Swiatlowod
                    where sw.IDnext == licz
                    select new { sw };

                x = szukaj.First().sw.IDS;
                if ((x == 0) || (x == pop))

```

```

        {
            flaga = 1;          //czyli nie znaleziono
            //wroc do foreach w poszukiwaniu nastepnego
        }
        else if ((x == source) && (po.siec.j >= dl))
        {
            k = new Ending(po.siec.IDS, po.siec.Pop, o.siec.j);
            flaga = 2;
            return po.siec.IDS;
        }
        licz = x;
    }
}

}
return 0;
}

```

D4. Ładowanie danych z bazy do źródła danych obiektu

```

DataSet ds = new DataSet();

string sql = "SELECT * FROM Ulica WHERE IDM = " + Login.IDmiejscowosc.ToString();
string connStr = "Data Source=localhost;Initial Catalog=siec;Integrated Security=True";
SqlConnection conn = new SqlConnection(connStr);
SqlDataAdapter dab = new SqlDataAdapter(sql, conn);
dab.Fill(ds, "Ulica");

BindingSource uBindingSource = new BindingSource();
uBindingSource.DataSource = ds;
uBindingSource.DataMember = "Ulica";

streetABComboBox.DataSource = uBindingSource;

streetABComboBox.DisplayMember = "Nazwa";
streetABComboBox.ValueMember = "IDU";
this.addBlockPanel.Visible = true;

```

Opis zawartości płyty CD

Nieodłączną częścią niniejszej pracy jest płyta CD zawierająca:

- tekst pracy w formacie doc i pdf,
- kody źródłowe programów,
- wersja instalacyjna systemu,
- skrypt generujący bazę danych,
- raport danych wygenerowany w programie Enterprise Architect.