



POLITECHNIKA ŚLĄSKA  
WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

Projekt inżynierski

Projekt i implementacja systemu przydzielania miejsc w akademikach

Autor: Michał Niesler

Kierujący pracą: dr inż. Alina Momot

Gliwice, Styczeń 2011

# Spis treści

Wstęp.....	3
1. Analiza Tematu.....	5
1.1 Podstawowe informacje.....	5
1.2 Określenie wymagań.....	6
2. Wybór narzędzi programowych.....	9
2.1 Języki programowania i zapytań.....	9
2.2 Narzędzia programowe i technologie.....	10
3. Specyfikacja wewnętrzna.....	12
3.1 Diagram przypadków użycia.....	12
3.2 Model bazy danych.....	14
3.3 Procedury SQL.....	16
3.4 Klasy.....	16
4. Specyfikacja zewnętrzna.....	18
4.1 Wymagania sprzętowe oraz programowe.....	18
4.2 Opis Instalacji.....	18
4.3 Instrukcja obsługi.....	18
5. Testowanie i uruchamianie.....	32
5.1 Opis testowania.....	32
5.2 Napotkane problemy.....	32
Podsumowanie.....	34
Literatura.....	35
Dodatek A. Opis tabel bazy danych.....	36
Dodatek B. Fragment skryptu generującego bazę danych.....	38
Dodatek C. Fragmenty kodu źródłowego.....	39
C1. Przyznanie miejsca.....	39
C2. Sprawdzenie poprawności wypełnienia formularzu adresu.....	39
C3. Dodanie nowego adresu do bazy danych.....	40
Opis zawartości płyty cd.....	41

# Wstęp

W dzisiejszych czasach coraz większa część społeczeństwa posiada dostęp do internetu. W Polsce według badań GFK Polonia ze stycznia 2009 z sieci korzystało już ponad 17.3 miliona osób. Również coraz więcej danych przechowywanych jest w bazach danych tworzonych na potrzeby firm, przedsiębiorstw oraz różnych instytucji, takich jak np. NFZ. Internet można służyć pomocą przy realizacji codziennych spraw, takich jak zakupy, prowadzenie konta bankowego, a nawet umożliwia zdalną pracę. Instytucje państwowe idąc z duchem czasu również wprowadzają możliwość składania różnych dokumentów przez internet. Szkoły i uczelnie prowadzą internetową rekrutację.

Politechnika Śląska jako uczelnia techniczna, a w szczególności wydział Automatyki, Elektroniki i Informatyki powinien przeprowadzić w wprowadzaniu internetowej obsługi studentów. Obecnie istnieje już wiele rozwiązań ułatwiających uczącym się na Politechnice załatwianie wielu spraw. Są to System Obsługi Toku Studiów, System Specjalności i Przedmioty Obieralne, System Prace Dyplomowe służący do wyboru tematów. Przydatne są też strony wydziałowe oraz strona Politechniki Śląskiej, gdzie można znaleźć wiele potrzebnych informacji. Świetnym rozwiązaniem jest internetowy plan zajęć, gdzie łatwo i szybko można znaleźć kto, gdzie i kiedy ma zajęcia. W planach są kolejne ułatwienia dla studentów i pracowników Politechniki Śląskiej, takie jak elektroniczny indeks.

Celem tej pracy inżynierskiej jest implementacja systemu wspomagającego proces przydzielania miejsc w akademikach. Aplikacja ta pozwoliłaby uprościć pracę dziekanatów, automatyzując proces przydziału miejsc. Ułatwienia objęłyby również studentów, szczególnie tych którzy zaczynają dopiero studia, ponieważ wnioski mogłyby być składane za pośrednictwem internetu. Pozwoliłoby to również na zmniejszenie ilości przechowywanych dokumentów, gdyż wyeliminowałoby wnioski składane w formie papierowej. Dzięki temu systemowi każdy student mógłby otrzymywać pocztą elektroniczną informację o otrzymaniu miejsca zaraz po zatwierdzeniu decyzji przez komisję. Program pozwoliłby też na automatyczne generowanie odpowiednich dokumentów, jak np. decyzja o przydziale miejsca, która musi być w wersji papierowej.

Praca ta jest pisana na podstawie rozmów z pracownikami i studentami Politechniki Śląskiej, jednak docelowo będzie mieć możliwość obsługi dowolnej uczelni, posiadającej akademiki. System przydziału miejsc w akademikach będzie przechowywał wszelkie niezbędne dane w bazie danych opartej na serwerze Microsoft SQL Server.

Niniejsza praca inżynierska została podzielona na 5 rozdziałów. W pierwszym z nich

zostały przedstawione główne cele oraz założenia jakie musi spełniać tworzony system. Rozdział drugi zawiera krótki opis wykorzystanych podczas implementacji narzędzi, między innymi języków programowania i zapytań, programów wspomagających projektowanie aplikacji, wykorzystanych technologii oraz środowiska programistycznego. Kolejny rozdział dotyczy specyfikacji wewnętrznej. Są w nim umieszczone i opisane diagramy przypadków użycia i schemat struktury bazy danych. Opisane są również szczegóły implementacji. Czwarty rozdział poświęcony jest specyfikacji zewnętrznej. Podane są w nim wymagania jakie musi spełniać sprzęt, na którym będzie instalowana aplikacja. Krótko opisana jest instalacja systemu. W rozdziale tym znajduje się również instrukcja obsługi. W ostatnim rozdziale zamieszczone są informacje dotyczące testowania. Opisano w nim w jaki sposób i na jakim sprzęcie zostały przeprowadzone testy, jakie napotkano problemy oraz w jaki sposób je rozwiązano. Na końcu pracy znajduje się bibliografia oraz dodatki.

# 1. Analiza Tematu

W tym rozdziale zostaną przedstawione informacje dotyczące osiedla studenckiego Politechniki Śląskiej, uczelni oraz zasad przydziału miejsc w akademikach. W tej części pracy zostanie również przedstawione określenie wymagań, które musi spełniać system.

## 1.1 Podstawowe informacje

Politechnika Śląska jest jedną z większych uczelni w Polsce i największą w regionie. Posiada 12 wydziałów: 9 w Gliwicach, 2 w Katowicach i jeden w Zabrzu. Istnieje również jednostka pozawydziałowa w Rybniku. Według danych podanych na stronie Politechniki Śląskiej obecnie, tj. w roku akademickim 2010/2011, na wszystkich wydziałach uczelni studiuje ponad 30 tysięcy osób [1]. Wiele z nich jest z Górnośląskiego Okręgu Przemysłowego. Jednak również wielu studentów jest przyjezdnych. Studia na Politechnice Śląskiej wybierają ludzie z całej Polski, a nawet z zagranicy. Z myślą o przyjezdnych zostały wybudowane domy studenckie. Obecnie uczelnia posiada 13 akademików, z tego aż 11 znajduje się w Gliwicach (Rzepicha, Piast, Ziemowit, Barbara, Ondraszek, Strzecha, Elektron, Karolinka, Karlik, Solaris), po jednym w Katowicach (Babilon), Zabrzu (Alaska) i Rybniku (Jedynaczek). Według informacji podanych na stronie Politechniki łącznie w domach studenckich może zamieszkać około 3500 studentów [1]. Każdy z wydziałów ma do dyspozycji określoną ilość miejsc dla swoich studentów. O podziale wszystkich miejsc pomiędzy wydziały decyduje rektorat.

Ponadto w domach studenckich po kilka miejsc mają do swojej dyspozycji Rektor Politechniki Śląskiej oraz Uczelniany Związek Samorządu Studenckiego. Miejsca te przyznawane są na podstawie osobnych wniosków. Miejsce w domu studenckim jest zakwalifikowane jako pomoc socjalna. Obecnie przydział miejsc odbywa się na podstawie pisemnych wniosków składanych w dziekanatach poszczególnych wydziałów. Wnioski te są rozpatrywane przez komisję, w skład której wchodzi prodziekan do spraw socjalnych, pracownik dziekanatu do spraw akademików oraz 3 osoby z samorządu studenckiego. Głównym kryterium, według którego podejmowana jest decyzja, jest odległość miejsca zameldowania od uczelni. Ponadto, ze względu na kwalifikację jako pomoc socjalna, uwzględniany jest również dochód. Kolejnym czynnikiem brany pod uwagę jest czas dojazdu.

Przydział miejsc odbywa się w dwóch etapach. Pierwszy odbywa się w czerwcu. Jest

on przeznaczony dla studentów, którzy kontynuują naukę na Politechnice. Drugi etap odbywa się już po rekrutacji nowego rocznika. Przeznaczony on jest głównie dla studentów rozpoczynających studia. Na każdy z etapów przeznaczona jest pewna pula miejsc. Na przykład w przypadku wydziału Automatyki, Elektroniki i Informatyki, według informacji podanych przez pracownika dziekanatu, na pierwszy etap przypada 500 miejsc do podziału, a na drugi 150. Wszystko to sprawia, że przydział miejsc jest skomplikowanym i pochłaniającym dużo czasu przedsięwzięciem. Ponadto zawsze duża część osób z listy rezerwowej jest w stanie zawieszenia i nie wie czy dostanie miejsce w akademiku, czy ma szukać mieszkania. Dlatego system przydzielania miejsc w akademikach, byłby bardzo przydatny i usprawniłby cały proces. Taki system korzystny byłby również z punktu widzenia ochrony środowiska, gdyż wyeliminowałby potrzebę zużywania papieru na wnioski.

Istnieje również możliwość zamieszkania w domach studenckich w okresie przerwy letniej. Przeznaczona jest ona zarówno dla studentów Politechniki Śląskiej, jak i innych osób. Jednak przydział miejsc na okres wakacyjny odbywa się na zupełnie innych zasadach, niż przydział miejsc na rok akademicki. Głównym czynnikiem decydującym o przyznaniu miejsca jest termin zgłoszenia. O przyznaniu miejsca decyduje kierownik danego akademika.

## 1.2 Określenie wymagań

W tym podrozdziale opisane są wymagania, które musi spełnić projektowany system. Na podstawie tych wymagań będą tworzone odpowiednie funkcje i ograniczenia programu. System przeznaczony będzie dla uczelni wyższych posiadających własne miasteczka studenckie i obsługiwać go będą pracownicy uczelni oraz studenci. Wymagania dla przejrzystości zostały podzielone według usług dla poszczególnych użytkowników.

Wymagania ogólne:

- działanie na więcej niż jednym stanowisku roboczym,
- obsługa programu wykorzystując przeglądarkę internetową,
- 4 typy użytkowników:
  - Administrator – przeznaczony dla pracowników administracji centralnej uczelni,
  - Dziekanat – przeznaczony dla pracowników dziekanatów poszczególnych wydziałów,
  - Kierownik – przeznaczony dla kierowników akademików,
  - Student – przeznaczony dla studentów uczelni zainteresowanych złożeniem wniosku o miejsce w akademiku,

- budowa modułowa, z modułami odpowiadającymi użytkownikom,
- dostęp do modułu tylko za pomocą konta odpowiadającego mu użytkownika.

Rejestracja i logowanie:

- logowanie na podstawie loginu i hasła,
- konto użytkownika „Student” zakładane samodzielnie przez zainteresowaną osobę w oknie rejestracji lub przez pracownika dziekanatu na podstawie danych z wniosku złożonego w formie papierowej,
- konto użytkowników „Administrator”, „Dziekanat” i „Kierownik” zakładane tylko i wyłącznie przez administratora,
- jednoznaczna identyfikacja użytkownika i jego typu na podstawie loginu,
- załadowanie odpowiedniego modułu na podstawie typu użytkownika,
- możliwość zmiany hasła.

Moduł „Administrator” udostępnia następujące funkcje:

- dodawanie, usuwanie i edycja danych dotyczących wydziałów,
- dodawanie, usuwanie i edycja danych dotyczących akademików,
- ustalanie terminów składania i rozpatrywania wniosków oraz termin kwaterowania,
- określanie ilości miejsc w poszczególnych akademikach przeznaczonych dla każdego z wydziałów,
- dodawanie i usuwanie kont „Administrator”, „Dziekanat” i „Kierownik”.

Moduł „Student” udostępnia następujące funkcje:

- wprowadzanie i edycja danych osobowych studenta: imię, nazwisko, pesel, adres, numer indeksu, wydział, kierunek, rok studiów, telefon, email;
- składanie wniosku i sprawdzanie jego statusu,
- wycofanie wniosku lub rezygnacja z przyznanego miejsca,
- usuwanie konta wraz z wszystkimi danymi na temat studenta.

Moduł „Dziekanat” udostępnia następujące funkcje:

- dodawanie danych osób, które złożyły wniosek w formie papierowej (zarejestrowanie studenta jako nowego użytkownika, wpisanie danych, złożenie wniosku),
- przeglądanie listy złożonych wniosków dla danego wydziału sortowanej według zadanego kryterium,
- przeglądanie wszystkich danych dotyczących studenta,
- przyznanie miejsca w akademiku, przesunięcie do listy rezerwowej, odrzucenie

wniosku,

- przeglądanie oraz wydruk list osób, którym przyznano miejsce w akademiku lub zostały umieszczone na liście rezerwowej;
- przeglądanie listy osób, których wnioski zostały odrzucone;
- wysyłanie powiadomień o przyznaniu miejsca za pomocą poczty elektronicznej,
- podział miejsc przeznaczonych dla wydziału pomiędzy I i II etap przydzielania miejsc.

Moduł „Kierownik” udostępnia następujące funkcje:

- przeglądanie listy osób, którym przyznano miejsce w danym akademiku;
- przeglądanie szczegółowych danych studenta,
- drukowanie listy studentów, którym przyznano miejsce w danym akademiku.



## 2. Wybór narzędzi programowych

W rozdziale tym opisane są narzędzia programowe, pozwalające utworzyć diagramy UML, zaprojektować bazę danych czy zaimplementować aplikacje, które będą wykorzystane przy tworzeniu systemu przedzielania miejsc w akademikach. Jest tu również uzasadnienie wyboru konkretnych narzędzi, języków programowania, języków zapytań oraz środowiska programistycznego.

### 2.1 Języki programowania i zapytań

#### C#

Język ten jest stosunkowo nowym językiem. W książce „ASP.NET dla każdego” można przeczytać, że pochodzi on od języków C i C++ oraz jest podobny do Javy. Został opracowany przez Microsoft specjalnie dla platformy .NET i wspólnego środowiska uruchomieniowego CLR [2].

Ze stron portalu MSDN można się dowiedzieć o cechach tego języka między innymi o bardzo przydatnym mechanizmie odśmiecania pamięci (ang. garbage collection), dzięki któremu programista nie musi przejmować się niszczeniem obiektów, o właściwościach, o delegatach i zdarzeniach klas czyli odpowiednikach wskaźników na funkcje oraz o tym, że C# oferuje wsparcie wszystkich typów danych CTS, czyli wszystkich najważniejszych typów, wykorzystywanych w systemach komputerowych [3].

#### LINQ

Język jest częścią technologii .NET. Na stronach portalu MSDN można przeczytać iż umożliwia on zadawanie pytań na obiektach, gdzie baza danych i jej elementy traktowane są jak obiekty[4]. Składnie języka LINQ (Language Integrated Query) jest prosta i przypomina SQL. Wykorzystanie tego języka przyspiesza oraz ułatwia tworzenie zapytań w kodzie.

#### SQL

Według książki Ullmana „Podstawowy wykład z baz danych” jest to najbardziej popularny mechanizm definiowania poleceń zapytań i modyfikacji w relacyjnych systemach baz danych [5]. Stosowany jest do tworzenia i modyfikowania baz danych oraz do umieszczania i pobierania danych z baz danych. Wykorzystywany będzie głównie w pierwszej fazie tworzenia aplikacji w celu utworzenia bazy danych oraz wprowadzenia podstawowych danych i w fazie testowania w celu sprawdzania poprawności danych

zapisywanych i pobieranych z bazy przez aplikacje.

## **2.2 Narzędzia programowe i technologie**

Istnieje wiele programów ułatwiających pracę przy projektowaniu systemów informatycznych. Pozwalają one między innymi na wykonanie diagramów UML, tworzenie baz danych oraz implementacje aplikacji. Poniżej krótko opisane są narzędzia, które będą użyte przy tworzeniu systemu.

### **StarUML**

Jest to darmowe narzędzie używane do modelowania systemu informatycznego przy użyciu języka opisu UML [6]. (ang. Unified Modeling Language, Ujednolicony Język Modelowania). Za jego pomocą w łatwy sposób można wygenerować diagramy przypadków użycia, klas i komponentów. Graficzne przedstawienie wymagań systemu pozwala na łatwiejsze analizowanie i projektowanie go. O wyborze zdecydowała łatwość obsługi oraz znajomość tego programu przez autora pracy.

### **.NET Framework**

Platforma .NET opracowana została przez Microsoft. Z informacji zamieszczonych na stronach MSDN można się dowiedzieć, że obejmuje ona środowisko uruchomieniowe oraz biblioteki klas, które dostarczają podstawowe funkcjonalności dla aplikacji [7]. Udostępnia między innymi technologie ASP.NET (Active Server Pages) służącą do tworzenia dynamicznych aplikacji internetowych oraz ADO.NET (ActiveX Data Object) umożliwiającą łączenie aplikacji z bazą danych. Wybór tej technologii związany był z wcześniejszym wyborem języka programowania oraz ogólną znajomością i doświadczeniem w pracy z produktami firmy Microsoft przez autora pracy.

### **Microsoft SQL Server 2008**

Jest to jeden z popularniejszych systemów zarządzania bazą danych typu klient-serwer. Na stronach Microsoftu można przeczytać, że system ten zapewnia najwyższy poziom zabezpieczeń, niezawodności i skalowalności [8]. Pozwala na zredukowanie czasu i kosztu tworzenia aplikacji oraz zarządzania nimi przy wykorzystaniu zapytań LINQ oraz usług obiektowych technologii ADO.NET. Udostępnia też nowe typy danych m.in. przechowujące informacje o dacie i godzinie.

Z MS SQL Server zintegrowany jest MS SQL Server Management Studio. Stanowi on graficzną alternatywę dla wbudowanych narzędzi konfiguracyjnych. Powodem wybrania tego

zestawu narzędzi jest to że pozwala on na wygodne i przejrzyste zarządzanie bazami danych, a w szczególności ich tworzenie, modyfikację przechowywanych obiektów, wykonywanie zapytań oraz kopii zapasowych.

### **Microsoft Visual Studio .NET 2008.**

Jest to zestaw narzędzi programistycznych firmy Microsoft, który pozwala na szybsze i łatwiejsze tworzenie aplikacji opartych na platformie .NET. Dzięki zintegrowaniu wielu narzędzi upraszcza proces projektowania, implementacji i testowania oprogramowania.

Na stronach Microsoftu można przeczytać, że Visual Studio 2008 zapewnia dostęp do platformy .NET Framework 3.5, która udostępnia między innymi wcześniej opisane języki C# oraz LINQ, mechanizm IntelliSense, który jest formą automatycznego uzupełniania i pozwala na wyświetlanie list składowych danej klasy oraz obiektów i metod dostępnych w bieżącym kontekście [9]. Umożliwia też łatwe nawiązanie połączenia z bazą danych znajdującą się na serwerze. Zapewnia łatwy dostęp do dokumentacji, pomocy technicznej oraz aktualizacji oprogramowania poprzez usługę MSDN.

## 3. Specyfikacja wewnętrzna

W rozdziale tym można się zapoznać z wewnętrzną specyfikacją tworzonej aplikacji. Umieszczony został tu diagram przypadków użycia wraz z opisem, który pozwala na zapoznanie się z zadaniami przydzielonymi poszczególnym aktorom. Omówiony został model bazy danych na podstawie zamieszczonego schematu. Krótko scharakteryzowano klasy utworzone w systemie.

### 3.1 Diagram przypadków użycia

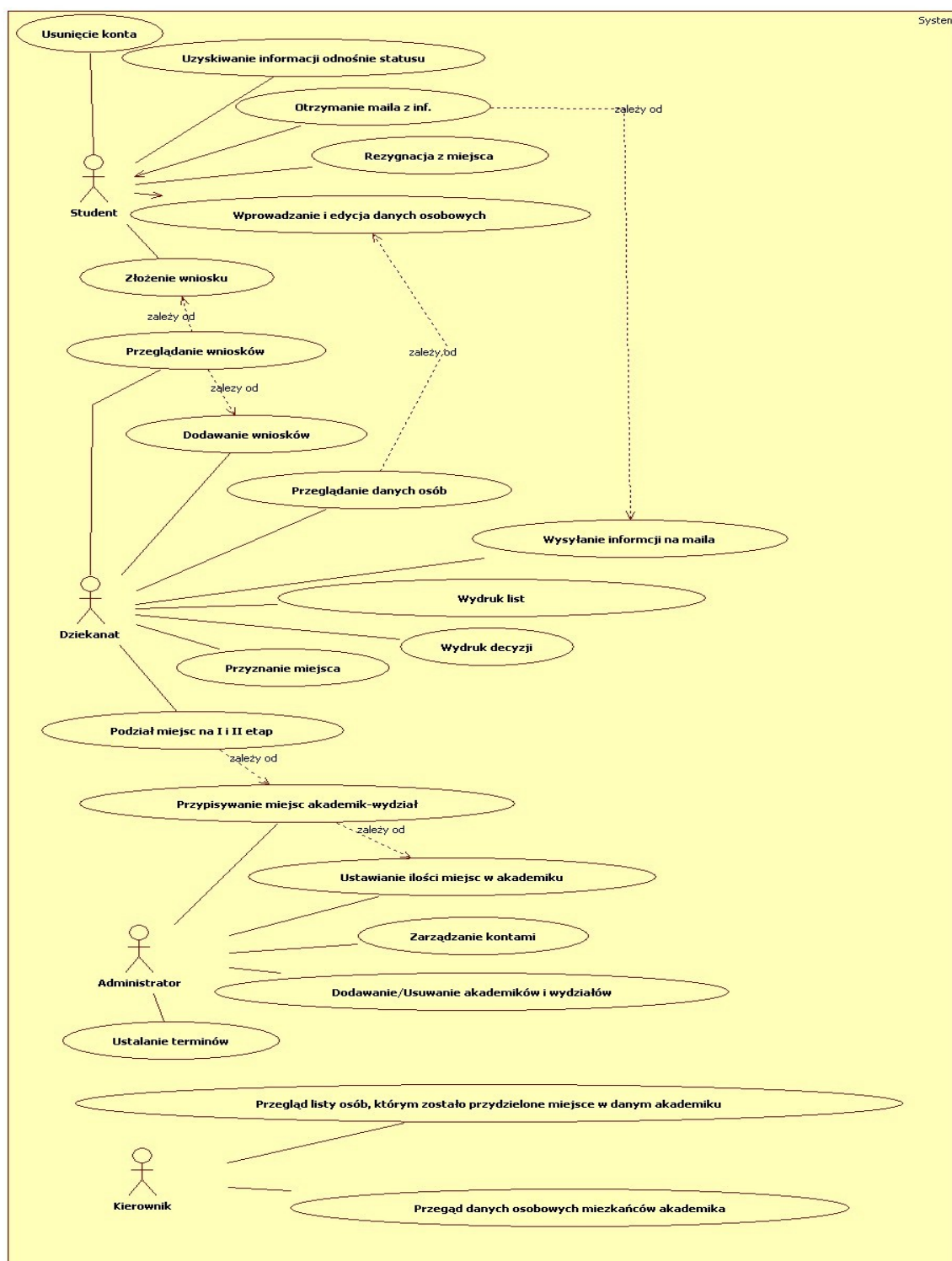
Diagram przedstawiony na rysunku 1 został wykonany za pomocą narzędzia StarUML. Diagram ten znajduje się również wśród załączników do pracy na płycie CD w pliku DiagramPrzypadkowUzycia.jpg. Na diagramie zaprezentowane są przypadki użycia dla systemu dla następujących aktorów: student, dziekanat, administrator, kierownik.

**Student** to użytkownik systemu, który ma dostęp do funkcji pozwalających na wpisanie i edycje danych osobowych potrzebnych do złożenia wniosku oraz jego wysłanie. Ponadto student może uzyskiwać informacje na temat statusu wniosku oraz w dowolnym momencie (nawet już po przyznaniu miejsca) zrezygnować z ubiegania się o miejsce. W przypadku przyznania miejsca student otrzymuje informację o tym fakcie pocztą elektroniczną. Użytkownik ten jako jedyny może samodzielnie usunąć własne konto.

**Dziekanat** to użytkownik systemu, który ma dostęp do funkcji umożliwiających przyznanie miejsca w akademiku, bądź odrzucenie wniosku. Osoba ta ma możliwość przeglądania danych osób z swojego wydziału, które złożyły wniosek. Dziekanat ma również możliwość dodawania wniosków osób, które taki wniosek złożyły w formie papierowej. Ponadto użytkownik ten może drukować decyzje o przyznaniu bądź nie przyznaniu miejsca, a także listy osób, którym przyznano i nie przyznano miejsca w akademikach oraz listę rezerwową. Dziekanat odpowiada też za podział miejsc przyznanych wydziałowi pomiędzy dwa etapy.

**Administrator** to użytkownik, który ma dostęp do funkcji dodawania i usuwania danych o akademikach oraz wydziałach. Może również modyfikować te dane, między innymi ilość miejsc przypisanych dla wydziału oraz ogółem w akademiku. Ponadto administrator odpowiada za zarządzanie kontami, może je dodawać lub usuwać. Jednak nie może usunąć własnego konta. Użytkownik ten odpowiada też za ustalanie terminów składania dokumentów, przydziału miejsc i kwaterowania.

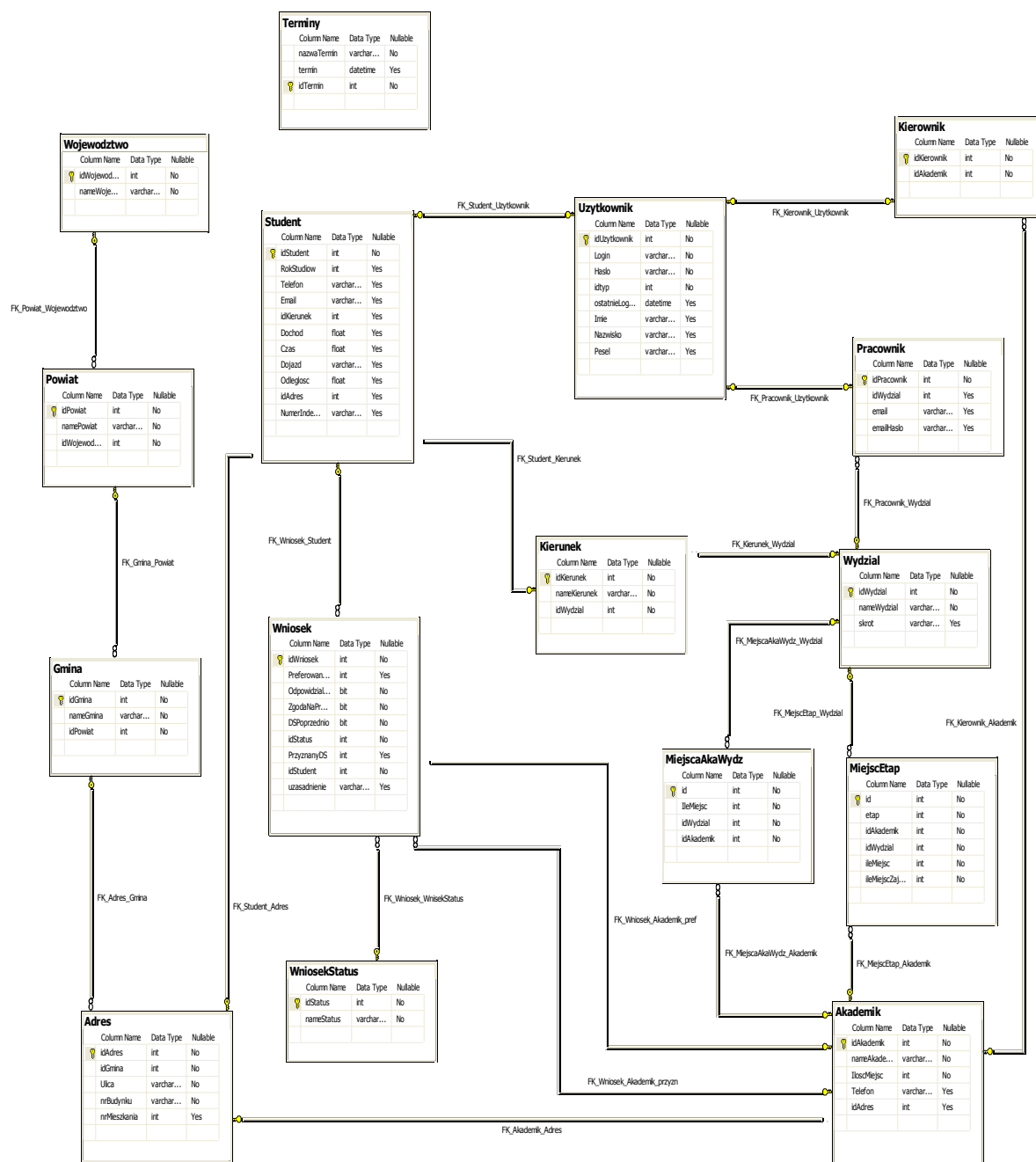
**Kierownik** to użytkownik, który ma dostęp do najmniejszej ilości funkcji w systemie. Osoba ta może przeglądać listę studentów przydzielonych do akademika, za który odpowiada oraz ma wgląd w ich dane. Posiada również możliwość wydruku tej listy wraz z niektórymi danymi.



Rysunek 1. Diagram przypadków użycia.

### 3.2 Model bazy danych

Schemat przedstawiony na rysunku 2 został utworzony przy użyciu SQL Server Management Studio. Rysunek ten znajduje się również wśród załączników do pracy na płycie CD w pliku SchematBD.jpg. Model przedstawia tabele wraz z kluczami podstawowymi. Rysunek przedstawia również relacje pomiędzy tabelami.



Rysunek 2. Schemat bazy danych.

Krótki opis wszystkich tabel bazy danych wraz z wyróżnionymi kluczami znajduje się w tabeli 1. Dokładniejszy opis tabel znajduje się w dodatku A niniejszej pracy.

Tabela 1. Opis tabel bazy danych.

Nazwa	Przechowywane dane	Klucze
Województwo	nazwy wszystkich województw	idWojewodztwo – podstawowy
Powiat	nazwy wszystkich powiatów	idPowiat – podstawowych, idWojewodztwo – obcy
Gmina	nazwy wszystkich gmin	idGmina – podstawowy, idPowiat – obcy
Adres	adresy akademików i miejsc zamieszkania studentów	idAdres – podstawowy, idGmina – obcy
Uzytkownik	podstawowe informacje o użytkownikach potrzebne m.in. do zalogowania	idUzytkownik – podstawowy
Kierownik	powiązania pomiędzy użytkownikiem typu kierownik, a akademikiem	idKierownik – podstawowy, obcy idAkademik – obcy
Pracownik	powiązania pomiędzy użytkownikiem typu dziekanat, a wydziałem oraz dane dodatkowe	idPracownik – podstawowy, obcy, idWydzial – obcy
Student	informacje o studentach potrzebne do złożenia wniosku oraz dane kontaktowe	idStudent – podstawowy, obcy, idAdres, idKierunek - obcy
Wniosek	informacje związane z wnioskiem, m.in. zgoda na przetwarzanie danych osobowych	idWniosek – podstawowy, idStudent, PreferownyDS, PrzyznanyDS, idStatus – obcy
Kierunek	nazwy wszystkich kierunków	idKierunek – podstawowy, idWydzial – obcy
Wydzial	informacje dotyczące wydziałów wydziałów	idWydzial – podstawowy
Akademik	informacje dotyczące akademików	idAkademik – podstawowy, idAdres – obcy
MiejscaAka Wydzial	ilość miejsc przeznaczonych dla wydziałów w poszczególnych akademikach	id – podstawowy, idWydzial, idAkademik - obcy
MiejscaEtap	ilość miejsc przeznaczona na I i II etap	id – podstawowy, idWydzial, idAkademik – obcy
Terminy	daty obowiązujące w procesie przydzielaniu miejsc	idTermin – podstawowy
WniosekStatus	możliwe statusy wniosku	idStatus – podstawowy

## 3.3 Procedury SQL

Do pobierania niektórych danych z bazy, zostało napisanych kilka procedur składowanych. Ich implementacja znajduje się w skrypcie tworzącym bazę danych, którego fragment znajduje się w dodatku B. Procedury wykorzystywane są przy tworzeniu raportów przeznaczonych do wydruku. Raporty te są listami studentów spełniających określone przez parametry procedur warunki. Opis procedur znajduje się w tabeli 2.

Tabela 2. Opis procedur SQL.

Nazwa	Opis	Parametry
lista	pobieranie danych dotyczący przyznanych miejsc w akademikach na danym wydziale, pobierane dane to imię, nazwisko, akademik w którym przyznano miejsce	IdWydział – identyfikator wydziału, którego dane mają dotyczyć
lista2	pobieranie danych osób z danego wydziału, których wnioski ma określony typ	IdWydział – identyfikator wydziału, IdTyp – numer typu statusu
lista3	pobieranie danych osób, którym zostało przydzielone miejsce w określonym akademiku	IdAkademik

## 3.4 Klasy

Język C# jest językiem obiektowym, dlatego wszystkie aplikacje pisane w tym języku są oparte na klasach. Podczas tworzenia zostało zaimplementowane około 50 klas. W aplikacji można rozróżnić kilka rodzajów klas. Fragmenty kodu zaimplementowanych klas można znaleźć w dodatku C.

Klasy dziedziczące po klasie `System.Web.UI.Page` są to główne klasy interfejsu. Łączą się one ze skryptem pozwalającym wygenerować stronę przez przeglądarkę. Klasa `Index` odpowiada za logowanie oraz rejestrację nowych użytkowników typu student. Pozostałe klasy tego typu, czyli `ModulAdministrator`, `ModulDziekanat`, `ModulKierownik`, `ModulStudent`, pozwalają poruszać się pomiędzy opcjami dostępnymi dla określonych użytkowników. Ustawiają parametry kontrolek opcji w zależności od terminów. Posiadają również zabezpieczenie przed nieautoryzowanym dostępem.

Klasy dziedziczące po klasie `System.Web.UI.UserControl` są to również klasy interfejsu, łączą się ze skryptem pozwalającym wygenerować na stronie kontrolkę. Klasy te



służą do komunikacji pomiędzy użytkownikiem, a aplikacją. Fragment kodu powodującego reakcję aplikacji na przycisk zamieszczony jest w dodatku C1. Zaimplementowane klasy to formularze, dzięki którym można wprowadzać dane oraz panele i tabele wyświetlające informacje pobierane z bazy danych. Odpowiadają za kontrolę poprawności formatu wprowadzanych danych. Fragment kodu sprawdzającego poprawność znajduje się w dodatku C2. Klasy te umożliwiają dostęp do poszukiwanych informacji. Przekazują dane do obiektów klas, które umożliwiają komunikację z bazą danych, za pomocą właściwości tych obiektów. Do tego rodzaju zaliczają się również klasy umożliwiające wydruk raportów z danymi.

Klasy umożliwiające komunikację z bazą danych. Pozwalają na dodawanie nowych rekordów w tablicach, ich usuwanie oraz modyfikacje. Fragment kodu wprowadzający informacje do bazy danych znajduje się w dodatku C3. Klasy te odpowiadają też za pobieranie informacji z rekordu. Obiekty tych klas obsługują wymianę danych pomiędzy interfejsem, a bazą danych. Niektóre z tych klas mają również dodatkowe funkcje. Przykładem jest klasa `UzytkownikClass`, która umożliwia sprawdzenie poprawności podanego przy logowaniu loginu oraz hasła.

W aplikacji utworzone zostały również inne klasy, których nie można zaliczyć do żadnej z powyższych kategorii. Najważniejszą z nich jest klasa `SPMData`. Za jej pomocą udostępniana jest struktura bazy danych, z której korzystają wszystkie klasy obsługujące wymianę danych pomiędzy aplikacją a bazą. System posiada również klasy obsługujące raporty, która odpowiadają za odpowiednie rozmieszczenie danych na wydruku, oraz klasę szyfrującą pozwalającą na zaszyfrowanie haseł użytkowników.

## 4. Specyfikacja zewnętrzna

Rozdział ten poświęcony jest opisowi specyfikacji zewnętrznej tworzonego systemu. Omówione zostały w nim między innymi wymagania sprzętowe oraz programowe, opis instalacji oraz ogólna instrukcja obsługi oprogramowania.

### 4.1 Wymagania sprzętowe oraz programowe

System Przydziału Miejsc do sprawnego działania wymaga minimum komputera z procesorem 1.2GHz oraz 512 MB pamięci operacyjnej RAM. Zalecany jest jednak komputer z procesorem 2GHz oraz 1GB pamięci RAM. System do działania wymaga zainstalowania następujących programów:

- SQL Server 2008,
- .Net Framework 3.5.

### 4.2 Opis Instalacji

Instalację aplikacji należy rozpocząć od utworzenia bazy danych. Baza ta powinna być utworzona na tym samym serwerze, na którym będzie umieszczona aplikacja. W tym celu administrator systemu bazy danych powinien wywołać następujące skrypty:

- SPMDDataCreate.sql – tworzy bazę danych oraz użytkownika systemu zarządzania bazą danych, za pomocą którego aplikacja będzie obsługiwała bazę,
- CreateTable.sql – tworzy tabele w bazie, widoki i procedury SQL;
- InsertDanePodstawowe – wprowadza do bazy podstawowe dane takie, jak np. województwa, powiaty, gminy, nazwy statusów wniosków;
- InsertDaneDodatkowe – wprowadza do bazy dane dodatkowe dotyczące politechniki (wydziały, kierunki, akademiki).

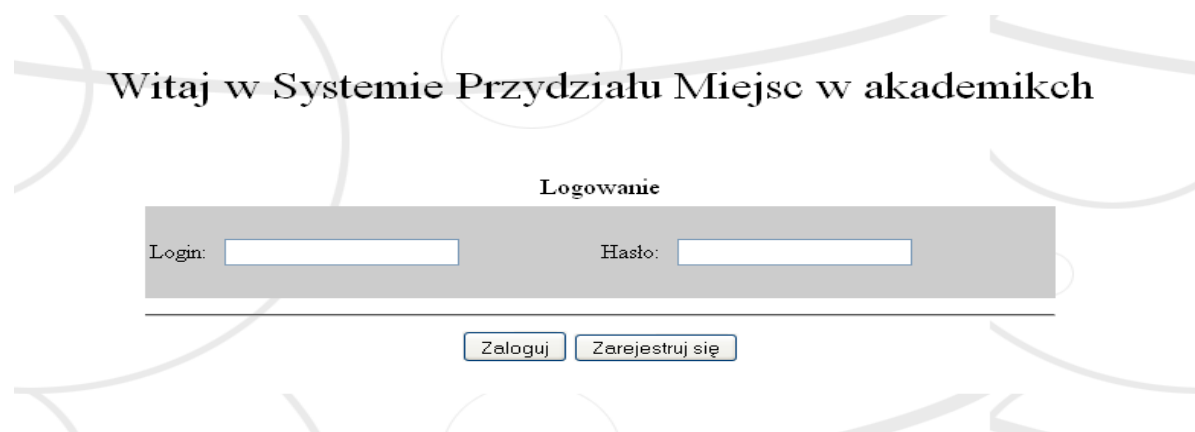
Następnie należy umieścić folder „Publish SPM” na serwerze stron www. Folder ten zawiera wszystkie pliki aplikacji potrzebne do działania strony.

### 4.3 Instrukcja obsługi

System składa się z czterech modułów: "Administrator", "Dziekanat", "Kierownik" i "Student". Każdy z tych modułów odpowiada jednemu typowi użytkownika. Tylko dany typ użytkownika może korzystać z danego modułu. Konto typu "Administrator" przeznaczone jest dla administracji centralnej oraz głównego administratora systemu. Konto typu

"Dziekanat" przeznaczone jest dla pracowników dziekanatów poszczególnych wydziałów. Konto typu "Kierownik" przeznaczone jest dla kierowników poszczególnych akademików. Te trzy konta zakładane oraz usuwane są tylko i wyłącznie przez administratora. Natomiast konto typu "Student" może być zakładane samodzielnie przez studenta, bądź przez pracownika dziekanatu po złożeniu przez studenta wniosku o przyznanie miejsca w akademiku w formie papierowej.

Obsługa „Systemu Przydziału Miejsc” (SPM) odbywa się przy wykorzystaniu przeglądarki internetowej. Praca z aplikacją rozpoczyna się od okna logowania, które przedstawione jest na rysunku 3. Aby się zalogować należy wpisać swój login oraz hasło w odpowiednie pola. Po zalogowaniu w zależności od typu konta, ładowana jest strona odpowiedniego modułu. Standardowo po instalacji oprogramowania jest założone tylko jedno konto o loginie „Administrator” i hasle „a”. Jest to konto przeznaczone dla głównego administratora. Osoba taka powinna się zalogować i zmienić hasło na własne. Funkcje dostępne dla administratora oraz pozostałych użytkowników przedstawione będą w dalszej części instrukcji.



Rysunek 3. Obszar logowania.

Widok ogólny stron poszczególnych modułów jest podobny. Widok strony powitalnej modułu "Administrator" przedstawiony jest na rysunku 4.

Podzielony jest ona na 4 obszary:

- menu - dzieli się na listę rozwijalną „Nawigacja”, w której znajdują się opcje obsługi systemu, listę rozwijalną dokumenty, w której znajdują się ewentualne dokumenty do druku (nie każdy moduł ma na tej liście opcje do wyboru) oraz opcję „Wyloguj” pozwalający na wylogowanie użytkownika i powrót do okna logowania,

- obszar roboczy – w nim znajdują się wszelkie formularze i kontrolki pozwalające na umieszczanie i pobieranie danych,
- obszar z logiem uczelni,
- obszar przeznaczony na informacje.

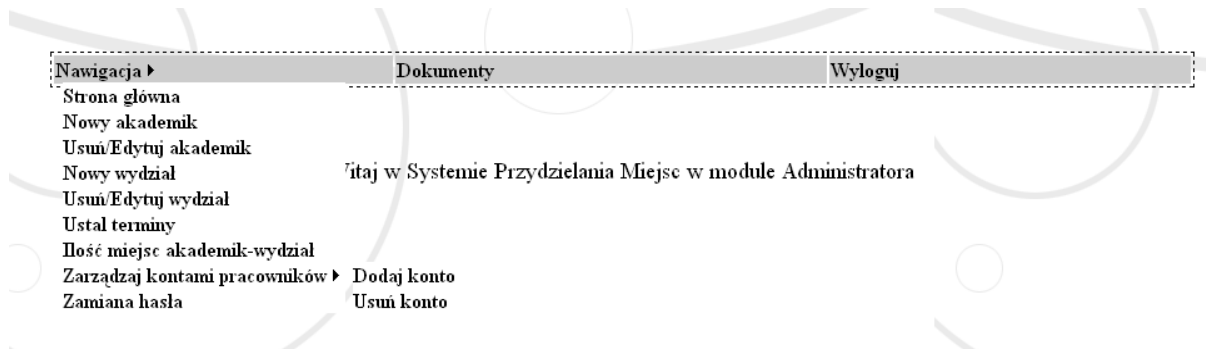


Rysunek 4. Okna przeglądarki z otwartą stroną powitalną Systemu Przydzielania Miejsc modułu "Administrator".

Każdy z modułów posiada w menu „Nawigacja” opcję „Zmiana hasła”. Aby zmienić hasło należy w odpowiednich polach wpisać hasło aktualnie używane oraz dwa razy nowe hasło i za pomocą przycisk „Zmień” zatwierdzić zmianę. W razie jakiś nieprawidłowości poniżej przycisku zostaną wyświetlone odpowiednie komunikaty.

### Użytkownik typu "Administrator"

Wygląd i opcje dostępne w menu „Nawigacja” dla tego typu użytkownika przedstawione są na rysunku 5. Zostaną one omówione w kolejności od góry.



Rysunek 5. Menu w module "Administrator".

**Strona główna** – pozwala na wyświetlenie komunikatu powitalnego.

**Nowy akademik** – umożliwia dodanie informacji o nowym akademiku. Po jej wybraniu pokaże się formularz, w którym należy uzupełnić dane na temat nowego akademika, takie jak nazwa, telefon, ilość miejsc oraz dane adresowe. Formularz ten przedstawiony jest na rysunku 6. Aby zapisać informacje o nowym akademiku należy użyć przycisku „Dodaj akademik”. Jeśli wszystkie pola są wypełnione dane akademika zostaną dodane do bazy danych. W przeciwnym razie obok pól nie wypełnionych pojawi się czerwona gwiazdka z informacją o braku wypełnienia.

The image shows a web form titled "Nowy akademik". At the top, there is a navigation bar with links: "Nawigacja", "Dokumenty", and "Wyloguj". Below the navigation bar, the form is divided into two main sections. The first section, titled "Nowy akademik", contains three input fields: "Nazwa akademika:", "Telefon:", and "Ilość miejsc:". The second section, titled "Adres akademika", contains three dropdown menus for "Województwo:", "Powiat:", and "Gmina:", and two text input fields for "Ulica:" and "Numer domu:". At the bottom of the form, there is a button labeled "Dodaj akademik".

Rysunek 6. Formularz dodawania informacji o nowym akademiku.

**Usuń/Edytuj akademik** – pozwala na usunięcie informacji dotyczących akademika, bądź edycję jego danych. Widok formularza opcji przedstawiony jest na rysunku 7. Na początek musimy wybrać akademik z listy. Pola automatycznie wypełniane są danymi o wybranym akademiku. Aby usunąć informacje o wybranym akademiku, należy użyć przycisku „Usuń akademik”. Aby dokonać zmian danych dotyczących akademika, należy zaznaczyć, które pola mają być zmieniane, a następnie wpisać nowe informacje. Zmiana zostanie wprowadzona dopiero po użyciu przycisku „Zatwierdź zmiany”.

Nawigacja ▶
Dokumenty
Wyloguj

Usun/Edytuj Akademik

Wybierz akademika: Barbara ▼
Usuń akademika

---

Edycja akademika

Nowa nazwa akademika
☐ Barbara

Nowa ilość miejsc
☐ 250

Nowy telefon
☐

☒ Zmiana Adresu

---

Województwo: śląskie ▼
Ulica: M.C.Skłodowskiej

Powiat: Gliwice ▼
Numer domu: 7

Gmina : Gliwice ▼

Zatwierdź zmiany

Rysunek 7. Formularz usuwania i edycji informacji dotyczących akademika.

**Nowy Wydział** – umożliwia dodanie informacji odnośnie nowego wydziału. Formularz opcji przedstawiony jest na rysunku 8. Aby dodać dane nowego wydziału wystarczy wpisać nazwę. Aby dodać nazwy kierunków otwartych na nowym wydziale trzeba wpisać ją oraz użyć przycisku „Dodaj kierunek”. W przypadku gdy użyjemy przycisku, a nazwa nie będzie wpisana pojawi się czerwona gwiazdka z informacją o braku wpisanej nazwy. W otwartych kierunkach można zmieniać nazwy używając przycisku „Zmień nazwę” lub usuwać je za pomocą przycisku „Usuń kierunek”. Zapisanie danych wydziału oraz kierunków odbędzie się dopiero po wybraniu przycisku „Dodaj wydział”, jeśli nazwa wydziału została wpisana. W przeciwnym razie pojawi się czerwona gwiazdka z odpowiednią informacją.

Nawigacja ▶
Dokumenty
Wyloguj

Nowy wydział

Nazwa wydziału: 
Skrót:

Nazwa kierunku : 

Dodaj kierunek
Zmień nazwę
Usuń kierunek

Dodaj wydział

Rysunek 8 Formularz dodawania informacji o nowym wydziale.

**Usuń/Edytuj wydział** – pozwala na usunięcie wydziału, bądź edycje jego danych. Obsługa tej opcji odbywa się podobnie jak opcji „Usuń/Edytuj akademik”, jej formularz przedstawiony jest na rysunku 9. Zmiany dotyczące kierunków dokonywane są w momencie użycia jednego z trzech przycisków dotyczących kierunków.

Rysunek 9. Formularz usuwania i edycji danych dotyczących wydziału.

**Ustal terminy** – opcja ta pozwala na ustalenie terminów. Formularz zmiany dat przedstawiony jest na rysunku 10. Jeśli terminy są ustalone, to pola są wypełnione odpowiednimi datami. Aby ustalić poszczególne terminy, należy zaznaczyć, który termin chcemy ustalić, a następnie wybrać na kalendarzu interesującą nas datę. Aby zatwierdzić należy użyć przycisku „Zapisz zmiany”. Zmiany terminów zostaną zatwierdzone, jeśli wszystkie pola będą wypełnione i każdy następny od góry termin będzie później niż poprzedni. W przeciwnym wypadku wyświetli się czerwona gwiazdka z informacją o błędzie w danym polu. Istnieje możliwość usunięcia wszystkich terminów za pomocą „Usuń terminy”. Wykorzystując ten przycisk usuwane są również wszystkie wnioski z bazy.

Rysunek 10. Formularz zmiany dat terminów.

**Ilość miejsc akademik-wydział** – umożliwia przypisanie określonej ilości miejsc w danym akademiku przypadających na dany wydział. Formularz opcji przedstawiony jest na rysunku 11. Aby przypisać miejsca należy wybrać z list wydział i akademik. W prawym dolnym polu tekstowym należy wpisać ilość miejsc. Powyżej jest pokazana ilość miejsc dostępnych w danym akademiku. Liczba wpisana musi być mniejsza od tej liczby. Jeśli będzie większa przy próbie zapisu za pomocą przycisku „Zapisz zmianę” obok tego przycisku pojawi się czerwona gwiazdka z odpowiednią informacją.

Rysunek 11. Formularz przypisywania ilości miejsc w akademiku przypadających na dany wydział.

**Zarządzanie kontami pracowników → Dodaj konto** – pozwala na dodawanie kont pracowników. Formularz opcji przedstawiony jest na rysunku 12. Aby utworzyć nowego użytkownika, należy wpisać odpowiednie dane oraz wybrać typ konta. Brak wpisanych danych, użycie loginu, który już istnieje oraz różnica między hasłem, a potwierdzeniem hasła spowoduje wyświetlenie czerwonych gwiazdek z odpowiednimi komunikatami przy próbie dodania konta za pomocą przycisku „Utwórz konto”.

Rysunek 12. Formularz tworzenia kont.



**Zarządzanie kontami pracowników** → **Usuń konto** - umożliwia przeglądanie danych dotyczących pracowników oraz usunięcie ich kont. Wybranie typu użytkownika spowoduje wyświetlenie listy z odpowiednimi danymi. Aby usunąć konto, należy wybrać pracownika, za pomocą przycisku „Select” znajdującego się obok jego danych w wyświetlonej liście. Następnie użyć przycisku „Usuń konto pracownika”. Nad przyciskiem pojawi się pytanie „Czy jesteś pewny, że chcesz usunąć to konto?”, należy potwierdzić chęć usunięcia konta.

**Usuń konto pracownika**

Czy jesteś pewny, że chcesz usunąć to konto?

Imię:  Nazwisko:  Pesel:

☐ Administrator ☒ Dziekanat ☐ Kierownik

	Imię	Nazwisko	Pesel	Login	Wydział
<a href="#">Select</a>	Agnieszka	Nowak	76031822175	AEI	Automatyki, Elektroniki i Informatyki
<a href="#">Select</a>	Marta	Wiśniewska	67010518943	Budownictwo	Budownictwa
<a href="#">Select</a>	Monika	Adamczyk	67082213131	Elektro	Elektryczny

Rysunek 13. Fragment okna z listą pracowników umożliwiający usunięcie użytkownika

### **Użytkownik typu "Student"**

Użytkownik ten jest jedynym, który samodzielnie może założyć konto. Można tego dokonać przechodząc z widoku okna logowania, za pomocą przycisku „Zarejestruj się” do widoku rejestracji. Następnie należy wpisać login oraz dwukrotnie hasło i użyć przycisku „Utwórz konto”. Jeśli dane wprowadzone są poprawnie a proponowany login jest unikalny w ramach danych zawartych w bazie danych, to konto zostanie utworzone i nastąpi powrót do widoku logowania. Wygląd i opcje dostępne w menu "Nawigacja" dla tego typu użytkownika przedstawione są na rysunku 14. Zostaną one omówione w kolejności od góry.

**Strona główna** – pozwala na wyświetlenie komunikatu powitalnego, wyświetla również terminy.

Nawigacja ▶	Dokumenty	Wyloguj
-------------	-----------	---------

[Strona główna](#)  
[Dane osobowe](#)  
[Wniosek](#)  
[Status wniosku](#)  
[Zmiana hasła](#)  
[Usun konto](#)  
Witaj w Module Studenta SPM

Strona główna w Systemie Przydziału Miejsc

Terminy:

Składania wniosków przed I etapem przydzielania miejsc rozpoczyna się od: **2011-01-27**  
 Składania wniosków przed I etapem przydzielania miejsc kończy się: **2011-01-28**  
 Zakończenie I etapu (przydzielenie miejsc): **2011-01-30**  
 Składanie wniosków przed II etapem przydzielania miejsc: **2011-01-31**  
 Składania wniosków przed II etapem przydzielania miejsc kończy się: **2011-02-01**  
 Zakończenie II etapu (przydzielenie miejsc): **2011-02-02**  
 Kwaterowanie od: **2011-02-03**

Rysunek 14. Menu oraz panel powitalny w module "Student".

**Dane osobowe** – umożliwia wprowadzenie danych osobowych studenta. Formularz który to umożliwia przedstawiony jest na rysunku 15. Aby zapisać zmiany, należy wypełnić wszystkie pola (numer mieszkania nie jest wymagany), a następnie użyć przycisku „Zapisz”. W razie braku wypełnienia pól pojawi się czerwona gwiazdka przy niewypełnionym polu.

Nawigacja ▶	Dokumenty	Wyloguj
-------------	-----------	---------

**Dane osobowe**

Imię	<input type="text" value="Michał"/>	Numer indeksu:	<input type="text" value="148762"/>
Nazwisko:	<input type="text" value="Niesler"/>	Wydział:	<input type="text" value="Automatyki, Elektroniki i Infor"/>
Pesel :	<input type="text"/>	Kierunek:	<input type="text" value="Informatyka"/>
Telefon:	<input type="text" value="508203843"/>	Rok studiów:	<input type="text" value="4"/>
Email	<input type="text" value="michal.niesler@gmail.com"/>		

**Adres stałego miejsca zamieszkania**

Województwo:	<input type="text" value="śląskie"/>	Ulica:	<input type="text" value="Piechy"/>
Powiat:	<input type="text" value="Ruda Śląska"/>	Numer domu:	<input type="text" value="5"/>
Gmina :	<input type="text" value="Ruda Śląska"/>	Numer Mieszkania:	<input type="text" value="5"/>

Rysunek 15. Formularz danych osobowych.

**Wniosek** – pozwala na uzupełnienie danych niezbędnych we wniosku oraz wysłanie go. Formularz wniosku znajduje się na rysunku 16. Aby wysłać wniosek muszą być wypełnione wszystkie pola oraz zaznaczone dwie dolne kratki. Do wysyłania wniosku służy przycisk „Złóż wniosek”. Jest on dostępny tylko, gdy aktualna data jest odpowiednią do składania wniosków.

The screenshot shows a web form titled "Informacje dodatkowe wymagane do wniosku". At the top, there is a navigation bar with links: "Nawigacja", "Dokumenty", and "Wyloguj". The form contains the following fields and elements:

- Preferowany akademik:** A dropdown menu with "Piast" selected.
- Dochód miesięczny na jednego członka rodziny:** A text input with "4000" and a unit dropdown with "zł" selected.
- Przybliżony, łączny czas dojazdu w obie strony:** A text input with "4" and a unit dropdown with "h" selected.
- Odległość miejsca zamieszkania od Gliwic (w kilometrach):** A text input with "20" and a unit dropdown with "km" selected.
- Opis dojazdu (środek lokomocji skąd – dokąd - PKS, PKP, WPK, tramwaj):** A text area containing the text "pkp, kzk-gop".
- Legal declarations:** Three checkboxes with corresponding text:
  - ☐ Korzystałem z Domu Studenckiego w ubiegłych latach
  - ☐ Świadomy odpowiedzialności karnej w przypadku podania nieprawdziwych danych oświadczam, że przedstawione przeze mnie we wniosku informacje, są zgodne ze stanem faktycznym.
  - ☐ Wyrażam zgodę na przetwarzanie moich danych osobowych zgodnie z ustawą z dnia 29 sierpnia 1997 r. o ochronie danych osobowych (t.j. Dz.U. z 2002 r. Nr 101, poz. 926, z późn. zm.) oraz przepisami wewnętrznymi obowiązującymi w Politechnice Śląskiej, wyłącznie w celu i w zakresie niezbędnym do rozpatrzenia i realizacji wniosku o przyznanie pomocy materialnej. Wiem, że przysługuje mi prawo wglądu i poprawiania moich danych osobowych.
- Submit button:** A button labeled "Złóż wniosek" at the bottom center.

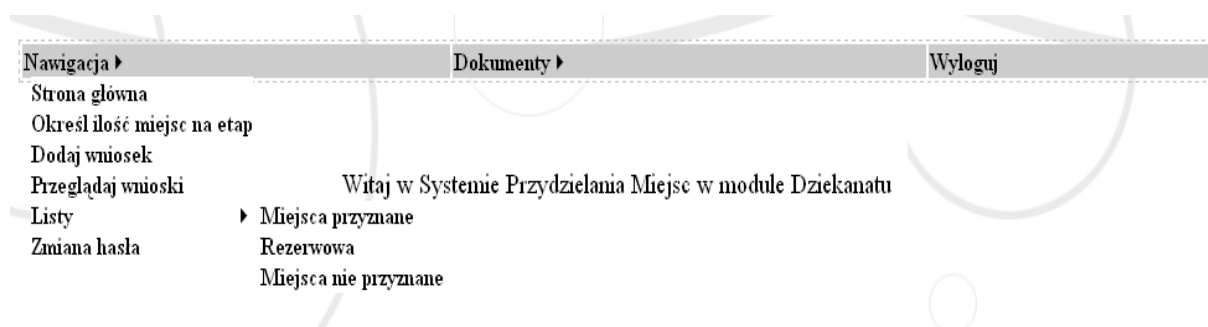
Rysunek 16. Formularz wniosku.

**Status wniosku** – podane są tu informacje na temat wniosku. Tutaj też można wycofać wniosek oraz zrezygnować z miejsca. Aby to zrobić, należy użyć przycisku „Wycofaj wniosek”

**Usuń konto** – umożliwia usunięcie konta. Aby to zrobić, należy potwierdzić chęć usunięcia konta.

### Użytkownik typu "Dziekanat"

Wygląd i opcje dostępne w menu "Nawigacja" dla tego typu użytkownika przedstawione są na rysunku 17. Zostaną one omówione w kolejności od góry.



Rysunek 17. Wygląd strony powitalnej i menu dostępne w module "Dziekanat"

**Strona główna** – pozwala na wyświetlenie komunikatu powitalnego.

**Określ ilość miejsc na etap** – umożliwia określenie ilości miejsc przypadających na I etap w danym akademiku przeznaczonych dla wydziału, w którym pracuje użytkownik. Formularz opcji przedstawiony jest na Rysunku 18. Ilość miejsc przeznaczonych na drugi etap określana jest automatycznie. Aby określić ilość miejsc należy wybrać z listy akademik, dla którego będziemy robili podział, a następnie wpisać liczbę w odpowiednie pole i wybrać przycisk „Zapisz zmianę”. Podana liczba musi być mniejsza lub równa, niż ta wyświetlana powyżej określająca ilość dostępnych miejsc. W przeciwnym razie zostanie wyświetlona czerwona gwiazdka.

Rysunek 18. Formularz pozwalający określić ilość miejsc przeznaczonych na I etap.

**Dodaj Wniosek** – pozwala pracownikowi na przepisanie wniosku papierowego. Formularz ten posiada takie same pola jak formularze z rysunków 15 i 16. Aby zapisać taki wniosek należy odpowiednio wypełnić wszystkie dane, zaznaczyć dwie ostatnie kratki, oraz użyć przycisku „Zapisz wniosek”. Złożenie wniosku jest równoznaczne z założeniem konta dla studenta o loginie imię.nazwiskoXXXX gdzie xxxx to cztery pierwsze cyfry peselu i hasło takim samym jak pesel. Wnioski można dodawać, tylko w odpowiednich terminach.

Nawigacja ▶

Dokumenty ▶

Wyloguj

Lista Wniosków

Szczegóły

Imię:

Nazwisko:

Pesel:

Numer indeksu:

Telefon:

Email:

Wydział:

Kierunek:

Rok:

Województwo:

Powiat:

Gmina:

Ulica:

Budynek:

Mieszkanie:

Odległość:

Czas dojazdu:

Dochód:

Opis dojazdu:

Preferowany akademik:

Korzystał z DS w poprzednich latach:

Wyświetlaj szczegóły

Akademik:

Ziemowit

Dostępne miejsca:

0

Przysuń miejsce

Przesuń do listy rezerwowej

Odrzuć wniosek

	Imię	Nazwisko	Odległość	Czas	Dochód	Preferowany akademik	DS Poprzednio
Select	Adam	Bułkowski	90	4	5433	Barbara	
Select	Marcin	Jasinski	50	5	900	Barbara	
Select	Michał	Niesler	20	4	4000	Piast	

Ilość wniosków:

3

Odśwież

Rysunek 19. Lista z złożonymi wnioskami wraz z panelem pozwalającym wyświetlić szczegółowe dane studentów.

**Przeglądaj Wnioski** - umożliwia przeglądanie danych z wniosków oraz przyznawanie miejsca, przesunięcie na listę rezerwową i odrzucenie wniosku. Widok listy wniosków i panel z danymi przedstawiony jest na rysunku 19. Dane dostępne są tylko w przypadku, gdy wniosek ma status rozpatrywany, czyli gdy etap składania wniosków jest zakończony. Aby wyświetlić szczegółowe dane dotyczące jednego wniosku, należy wybrać ten wniosek za

29

pomocą przycisku „Select” w pierwszej kolumnie tabeli oraz zaznaczyć kratkę „Wyświetlaj szczegóły”. Opcja ta udostępnia trzy przyciski „Przyznaj miejsce”, „Przesuń do listy rezerwowej” i „Odrzuć wniosek”. Pierwszy z nich dostępny jest, gdy trwa etap rozpatrywania wniosków i pozwala na przyznanie miejsca po uprzednim wybraniu akademika z listy i wniosku z tabeli. Drugi jest aktywny tylko w trakcie rozpatrywania wniosków w II etapie i umożliwia przesunięcie wniosku na listę rezerwową. Trzeci podobnie jak pierwszy dostępny jest w trakcie etapu rozpatrywania wniosków. Jego użycie powoduje odrzucenie wniosku. Na prawo od listy wyboru akademika znajduje się pole, w który podawana jest liczba jeszcze wolnych miejsc przeznaczonych dla danego akademika. Poniżej tabeli znajduje się licznik wniosków oraz przycisk odświeżający ją.

**Listy → Miejsca przyznane** – pozwala na przeglądanie danych osób, którym przyznano miejsce. Widok listy tych osób przedstawiony jest na rysunku 20. Aby wyświetlić dane szczegółowe, podobnie jak to było w opcji „Przeglądaj wnioski”, należy wybrać za pomocą „Select” osobę oraz zaznaczyć kratkę „Wyświetlaj szczegóły”.

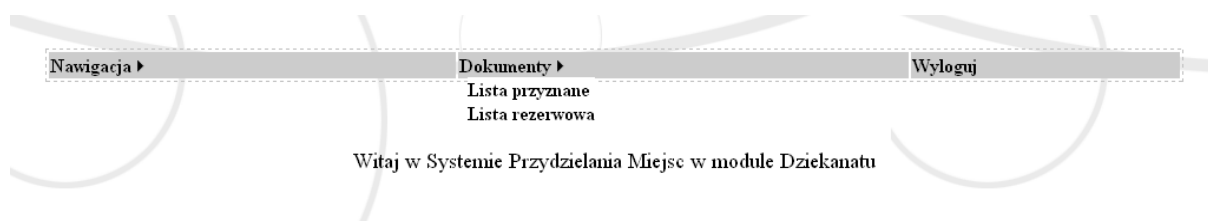
	Imię	Nazwisko	Pesel	Wydział	Kierunek	Rok	Numer Indeksu	Przyznany akademik
Select	Marta	Pabrol	44455555	Automatyki, Elektroniki i Informatyki	Informatyka	3	5455445	Ondraszek
Select	Monika	Sapel	3435534564	Automatyki, Elektroniki i Informatyki	Informatyka	3	53454	Barbara

Rysunek 20. Lista osób, którym zostało przyznane miejsce w akademiku.

**Listy → Rezerwowa** – umożliwia przeglądanie danych z wniosków, które są na liście rezerwowej, przyznawanie miejsca i odrzucenie wniosku. Wygląd i obsługa jest analogiczna jak w opcji „Przeglądaj wnioski”. Brakuje jedynie przycisku „Przesuń do listy rezerwowej”.

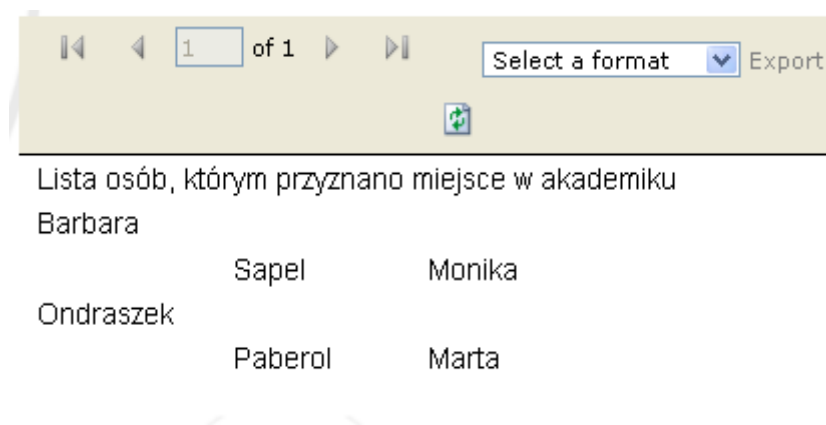
**Listy → Miejsca nieprzyznane** – Różni się od opcji „Miejsca przyznane” jedynie tym, że wyświetla osoby, którym nie przyznano miejsca.

Wygląd i opcje menu „Dokumenty” przedstawiony jest na rysunku 21 i zostaną one omówione w kolejności od góry.



Rysunek 21. Menu "Dokumenty".

**Lista przyznane** – generuje raport z listą studentów, którym przydzielono miejsce. Lista ta jest podzielona według akademików, w którym miejsce zostało przydzielone danemu studentowi. Za pomocą przycisku „Export” można wygenerować plik zawierający listę wybierając jeden z dostępnych formatów: \*.xls (pliki programu MS Excel) lub \*.pdf (pliki programu Adobe Reader). Okno raportu przedstawione jest na rysunku 22.



Rysunek 22. Raport z listą studentów, którym przyznano miejsce.

**Lista Rezerwowa** – generuje raport z listą studentów, którzy zostali umieszczeni na liście rezerwowej. Działanie i wygląd okna raportu jest analogiczne jak w przypadku raportu „Lista przyznane”.

### Użytkownik typu "Kierownik"

Użytkownik ten w menu "Nawigacja" posiada tylko trzy opcje: „Strona główna”, „Lista studentów do zakwaterowania” oraz „Zmiana hasła”. Dwie z nich zostały już opisane. Opcja „Lista studentów do zakwaterowania” natomiast działa podobnie jak opcje z modułu "Dziekanat": „Miejsca przyznane” oraz „Miejsca nieprzyznane” z tą różnicą, że wyświetla dane studentów, którym przyznano miejsce w akademiku powiązanych z danym użytkownikiem. W menu "Dokumenty" dostępna jest opcja "Lista Studentów", która pozwala wygenerować raport z listą studentów, którym przyznano miejsce w danym akademiku.

## 5. Testowanie i uruchamianie

Testowanie jest jednym z etapów tworzenia oprogramowania. W rozdziale tym opisany jest sposób w jaki testowany był System Przydziału Miejsc. Wyszczególnionych jest kilka napotkanych problemów oraz ich rozwiązanie.

### 5.1 Opis testowania

Testowanie przeprowadzane było w dwóch etapach i wykonywane wyłącznie przez autora niniejszej pracy. Pierwszy etap polegał na testowaniu równocześnie z tworzeniem kolejnych funkcji i modułów aplikacji. W tym etapie testowania wykorzystana była przeglądarka Mozilla Firefox w wersji 3.5. Wprowadzane były przykładowe dane i wykonywana operacja. Jeśli w trakcie operacji wystąpiły błędy, informacja o nich była najczęściej, w łatwy do przeanalizowania sposób, wyświetlana w oknie przeglądarki. Tym samym łatwo było zlokalizować błąd. W przypadku gdy błędy nie wystąpiły, sprawdzane były rezultaty w bazie danych. Jeśli rezultaty były niezgodne z założonymi rozpoczynało się szukanie błędu za pomocą debuggera. Drugi etap polegał na przeprowadzeniu testów gdy program był już gotowy. Pozwoliło to wykryć błędy spowodowane między innymi zmianami w strukturze bazy danych lub zmianami we fragmentach kodu, z których korzystały różne elementy systemu. Do testowania końcowego wykorzystane zostały przeglądarki Mozilla Firefox w wersji 3.5 oraz Opera w wersji 10.

Podczas testowania i szukania błędów wykorzystano mechanizm punktów kontrolnych (breakpoints) oraz mechanizm pracy krokowej. Testy były przeprowadzane na komputerze z procesorem Intel Core Duo 2.00GHz, 3GB pamięci RAM oraz systemem operacyjnym Windows XP Professional SP2.

### 5.2 Napotkane problemy

Większość napotkanych błędów i problemów dotyczyło komunikacji między interfejsem użytkownika lub samego interfejsu.

#### **Dodawanie kierunków przy tworzeniu nowego kierunku**

Początkowo dodawanie kierunków do bazy odbywało się od razu po wybraniu przycisku. Jednak w przypadku tworzenia nowego kierunku nie sprawdzało się to, gdyż pole idWydział w tabeli Kierunki nie przyjmowało wartości null. Rozwiązaniem było, w przypadku korzystania z opcji tworzenia nowego wydziału, utworzenie listy, w której



przechowywane są nazwy kierunków i dodanie do bazy tych nazw dopiero w momencie, gdy informacje o wydziale były już dodane.

### **Dodawanie nowych danych do tablic oraz wiązanie ich z innymi tablicami**

Problemem w tym przypadku okazało się automatyczne tworzenie identyfikatora elementu (klucza podstawowego). Rozwiązane zostało to na dwa sposoby: Dla części tablic, w których, występują inne unikalne pola jak na przykład w tablicy Uzytkownik pole Login, zostały napisane metody służące do pobierania identyfikatora poprzez przyrównanie wartości tych unikalnych pól. W pozostałych zamiast automatycznego tworzenia identyfikatora zastosowany został mechanizm wyszukania największego klucza podstawowego i dodanie do niego 1 w celu utworzenia nowego elementu. Wygenerowany w ten sposób identyfikator mógł być wpisany do powiązanej tabeli jako klucz obcy.

### **Nieautoryzowany dostęp do modułów**

Początkowo istniała możliwość dostępu do modułów mimo braku zalogowanego użytkownika. Aby dostać się do modułu wystarczyło wpisać jego adres w przeglądarce. Problem ten został wyeliminowany poprzez tworzenie zmiennej sesyjnej w trakcie logowania. Zmienna ta jest sprawdzana przy każdym ładowaniu modułów i jeśli nie jest zgodna z zadana wartością, osoba korzystająca z przeglądarki przekierowywana jest do okna logowania.

### **Brak kontroli typu wpisywanych danych**

Dane wpisywane w formularzach są traktowane jako ciąg znakowy. Użytkownik w miejscu, gdzie należało wpisać liczbę, ma możliwość wpisania innego znaku. Powodowało to błędy przy próbie zamiany ciągu znakowego na liczbę. Aby to wyeliminować został wstawiony blok try-catch, który próbuje dokonać takiej konwersji i w przypadku, gdy nie jest ona możliwa, powoduje wyświetlenie odpowiedniego komunikatu przy polu tekstowym, w którym podany jest nieprawidłowy ciąg.

### **Możliwość ustawienia kolejności dat niezgodnej z kolejnością terminów**

Przykładem tego błędu jest wpisanie daty rozpatrywania wniosków wcześniejszej niż daty składania wniosków. Rozwiązaniem tego problemu jest wstawienie sprawdzania każdych dwóch sąsiednich terminów w celu ustalenia czy data terminu, który powinien być wcześniej, nie jest późniejsza niż data terminu, który powinien być później. Sprawdzane jest to w momencie próby zapisu dat.

# Podsumowanie

Celem pracy inżynierskiej było utworzenie systemu wspomagającego proces przydzielania miejsc w akademikach przez uczelnie. Zrealizowana aplikacja pozwala na swobodne zmiany danych opisujących struktury organizacji uczelni (wydziały, kierunki) oraz zarządzanego przez nią miasteczka studenckiego. Umożliwia studentom złożenie wniosku o miejsce w akademiku oraz na bieżąco sprawdzanie stanu tego wniosku. Dzięki tej aplikacji pracownicy dziekanatów zajmujący się przydzielaniem w łatwy sposób mogą sortować i przeglądać wnioski. Przyspiesza to przepływ informacji pomiędzy wydziałami, a kierownikami akademików.

Utworzony w ramach tej pracy System Przydzielania Miejsc, mimo że powstał na podstawie rozmów z pracownikami Politechniki Śląskiej z powodzeniem może być stosowany przez inne uczelnie, szczególnie te większe posiadające własne miasteczka studenckie. Wprowadzenie nie wyeliminuje potrzeby przeglądania wniosków przez odpowiednie komisje, jednak pozwoli zaoszczędzić sporo czasu.

Jako potencjalne kierunki rozwoju systemu można wymienić zintegrowanie go z Systemem Obsługi Toku Studiów Politechniki Śląskiej oraz automatyzację procesu przydzielania miejsc. Integracja z platformą SOTS wyeliminowałaby potrzebę podwójnego przechowywania tych samych danych dotyczących studentów oraz pozwoliłaby studentom z jednego miejsca zarządzać swoimi sprawami dotyczącymi studiów. Natomiast automatyzacja przydzielania miejsc pozwoliłaby w jeszcze większym stopniu zaoszczędzić czas pracownikom dziekanatów.

# Literatura

- [1] Strona Politechniki Śląskiej, dostępna pod adresem [www.polsl.pl](http://www.polsl.pl) (styczeń 2011)
- [2] Chris P., ASP.NET dla każdego, Helion, Gliwice, 2002.
- [3] Dokumentacja języka C#, dostępna pod adresem <http://msdn.microsoft.com/en-us/library/kx37x362.aspx> (styczeń 2011)
- [4] Opis języka LINQ, dostępny pod adresem [http://msdn.microsoft.com/en-us/library/bb308959.aspx#linqoverview\\_topic1](http://msdn.microsoft.com/en-us/library/bb308959.aspx#linqoverview_topic1) (styczeń 2011)
- [5] Ullman J., Widom J., Podstawowy wykład z systemów baz danych", WNT, 2006.
- [6] Strona projektu StarUML – platformy UML/MDA typu open source, dostępna pod adresem <http://staruml.sourceforge.net/en/> (styczeń 2011)
- [7] Strona z opisem platformy .NET Framework  
<http://msdn.microsoft.com/library/zw4w595w.aspx> (styczeń 2011)
- [8] Strona produktu SQL Server, dostępna pod adresem  
<http://www.microsoft.com/sqlserver/2008/en/us/default.aspx> (styczeń 2011)
- [9] Dokumentacja środowiska Visual Studio 2008, dostępna pod adresem  
<http://msdn.microsoft.com/en-us/library/52f3sw5c%28v=VS.90%29.aspx> (styczeń 2011)

# Dodatek A. Opis tabel bazy danych

**Adres** (idAdres, idGmina, Ulica, nrBudynku, nrMieszkania)

idAdres (Typ: int, Rola: Primary Key, Opis: identyfikator adresu)

idGmina (Typ: int, Rola: Foreign Key, Opis: identyfikator określający gminę do której odnosi się adres)

Ulica (Typ: varchar(50), Opis: określa nazwę ulicy)

nrBudynku (Typ: varchar(10), Opis: określa numer budynku, uwzględnia litery)

nrMieszkania (Typ: int, Opis: określa numer mieszkania)

**Akademik** (idAkademik, idAdres, nameAkademik, IloscMiejsc, Telefon)

idAkademik (Typ: int, Rola: Primary Key, Opis: identyfikator akademika)

idAdres (Typ: int, Rola: Foreign Key, Opis: identyfikator określający adres akademika)

nameAkademik (Typ: varchar(50), Opis: określa nazwę akademika)

IloscMiejsc (Typ: int, Opis: określa ilość miejsc, jaką dysponuje akademik)

Telefon (Typ: varchar(20), Opis: określa numer telefonu akademika)

**Gmina** (idGmina, idPowiat, nameGmina)

idGmina (Typ: int, Rola: Primary Key, Opis: identyfikator gminy)

idPowiat (Typ: int, Rola: Foreign Key, Opis: identyfikator określający powiat, w którym znajduje się gmina)

nameGmina (Typ: varchar(50), Opis: określa nazwę gminy)

**Kierownik** (idKierownik, idAkademik)

idKierownik (Typ: int, Rola: Primary Key, Foreign Key, Opis: identyfikator kierownika oraz określający użytkownika)

idAkademik (Typ: int, Rola: Foreign Key, Opis: identyfikator określający akademik, którego użytkownik jest kierownikiem)

**Kierunek** (idKierunek, idWydzial, nameKierunek)

idKierunek (Typ: int, Rola: Primary Key, Opis: identyfikator kierunku)

idWydzial (Typ: int, Rola: Foreign Key, Opis: identyfikator określający wydział, na którym istnieje kierunek)

nameKierunek (Typ: varchar(50), Opis: określa nazwę kierunku)

**MiejscaAkaWydz** (id, idAkademik, idWydzial, IleMiejsc)

id (Typ: int, Rola: Primary Key, Opis: identyfikator określenia ilości miejsc)

idAkademik (Typ: int, Rola: Foreign Key, Opis: identyfikator określający akademik, którego dotyczy połączenie)

idWydzial (Typ: int, Rola: Foreign Key, Opis: identyfikator określający wydział, którego dotyczy połączenie)

IloscMiejsc (Typ: int, Opis: określa ilość miejsc, jaką przypisano wydziałowi w akademiku)

**MiejsceEtap** (id, idAkademik, idWydzial, etap, ileMiejsc, ileMiejscZajetych)

id (Typ: int, Rola: Primary Key, Opis: identyfikator określenia ilości miejsc na etap)

idAkademik (Typ: int, Rola: Foreign Key, Opis: identyfikator określający akademik, którego dotyczy połączenie)

idWydzial (Typ: int, Rola: Foreign Key, Opis: identyfikator określający wydział, którego dotyczy połączenie)

etap (Typ: int, Opis: określa etap)

iloscMiejsc (Typ: int, Opis: określa ilość miejsc, jaką przypisano wydziałowi w akademiku w danym etapie)

iloscMiejscZajetych (Typ: int, Opis: określa ilość miejsc, jaka została już zajęta)

**Powiat** (idPowiat, idWojewodztwo, namePowiat)

idPowiat (Typ: int, Rola: Primary Key, Opis: identyfikator powiatu)

idWojewodztwo (Typ: int, Rola: Foreign Key, Opis: identyfikator określający województwo, w którym znajduje się powiat)

namePowiat (Typ: varchar(50), Opis: określa nazwę powiatu)

**Pracownik** (idPracownik, idWydzial, email, emailHaslo)

idPracownik (Typ: int, Rola: Primary Key, Foreign Key, Opis: identyfikator pracownika oraz określający użytkownika)

idWydzial (Typ: int, Rola: Foreign Key, Opis: identyfikator określający wydział, którego użytkownik jest pracownikiem)

email (Typ: varchar(50), Opis: określa adres email pracownika)  
emailHaslo (Typ: varchar(50), Opis: określa hasło do maila pracownika, aby możliwe było wysłanie wiadomości)

**Student** (idStudent, idAdres, idKierunek, Telefon, Email, NumerIndeksu, Dojazd, Odleglosc, Czas, Dochod, RokStudiów)

idStudent (Typ: int, Rola: Primary Key, Foreign Key, Opis: identyfikator studenta oraz określający użytkownika)

idAdres (Typ: int, Rola: Foreign Key, Opis: identyfikator określający adres zameldowania studenta)

idKierunek (Typ: int, Rola: Foreign Key, Opis: identyfikator określający kierunek, na którym studiuje)

Telefon (Typ: varchar(20), Opis: określa numer telefonu studenta)

Email (Typ: varchar(50), Opis: określa adres email studenta)

NumerIndeksu (Typ: varchar(20), Opis: określa numer indeksu studenta)

Dojazd (Typ: varchar(255), Opis: określa opis dojazdu studenta na uczelnię)

Odleglosc (Typ: float, Opis: określa odległość miejsca zamieszkania studenta od uczelni)

Czas (Typ: float, Opis: określa czas dojazdu w obie strony studenta)

Dochod (Typ: float, Opis: określa dochód studenta w przeliczeniu na jednego członka w rodzinie)

RokStudiów (Typ: int, Opis: określa, na którym roku znajduje się student)

**Terminy** (idTermin, nazwaTermin, termin)

idTermin (Typ: int, Rola: Primary Key, Opis: identyfikator terminu)

nazwaTermin (Typ: varchar(50), Opis: określa nazwę terminu)

termin (Typ: datetime, Opis: określa datę terminu)

**Uzytkownik** (idUzytkownik, Login, Haslo, idtyp, OstatnieLogowanie, Imie, Nazwisko, Pesel)

idUzytkownik (Typ: int, Rola: Primary Key, Opis: identyfikator użytkownika)

Login (Typ: varchar(50), Opis: określa login użytkownika)

Haslo (Typ: varchar(50), Opis: określa hasło użytkownika, zaszyfrowane)

idtyp (Typ: int, Opis: określa typ użytkownika, co pozwala określić, do którego modułu ma dostęp)

OstatnieLogowanie (Typ: datetime, Opis: określa datę ostatniego logowania)

Imie (Typ: varchar(50), Opis: określa imię użytkownika)

Nazwisko (Typ: varchar(50), Opis: określa nazwisko użytkownika)

Pesel (Typ: varchar(50), Opis: określa pesel użytkownika)

**Wniosek** (idWniosek, PreferowanyDS, PrzyznanyDS, idStudent, idStatus, OdpowiedzialnoscKarna, ZgodaNaPrzetwDanych, DSPoprzednio, uzasadnienie)

idWniosek (Typ: int, Rola: Primary Key, Opis: identyfikator wniosku)

PreferowanyDS (Typ: int, Rola: Foreign Key, Opis: identyfikator określający akademik, w którym student chciałby zamieszkać)

PrzyznanyDS (Typ: int, Rola: Foreign Key, Opis: identyfikator określający akademik, w którym studentowi przyznano miejsce)

idStudent (Typ: int, Rola: Foreign Key, Opis: identyfikator określający studenta, którego dotyczy wniosek)

idStatus (Typ: int, Rola: Foreign Key, Opis: identyfikator określający status, jaki posiada wniosek)

OdpowiedzialnoscKarna (Typ: bit, Opis: określa potwierdzenie prawdziwości danych pod groźbą odpowiedzialności karnej)

ZgodaNaPrzetwDanych (Typ: bit, Opis: określa zgodę na przetwarzanie danych osobowych)

DSPoprzednio (Typ: bit, Opis: określa czy student mieszkał wcześniej w akademiku)

uzasadnienie (Typ: varchar(255), Opis: określa uzasadnienie decyzji dotyczącej wniosku)

**WniosekStatus** (idStatus, nameStatus)

idStatus (Typ: int, Rola: Primary Key, Opis: identyfikator statusu wniosku)

nameStatus (Typ: varchar(30), Opis: określa nazwę statusu)

**Wojewodztwo** (idWojewodztwo, nameWojewodztwo)

idWojewodztwo (Typ: int, Rola: Primary Key, Opis: identyfikator województwa)

nameWojewodztwo (Typ: varchar(50), Opis: określa nazwę województwa)

**Wydzial** (idWydzial, nameWydzial, skrot)

idWydzial (Typ: int, Rola: Primary Key, Opis: identyfikator wydziału)

nameWydzial (Typ: varchar(50), Opis: określa nazwę wydziału)

skrot (Typ: varchar(15), Opis: określa skrót nazwy wydziału)

# Dodatek B. Fragment skryptu generującego bazę danych

```

/***** Object: Table [dbo].[Wniosek]      Script Date: 01/20/2011 04:21:45 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Wniosek](
    [idWniosek] [int] IDENTITY(1,1) NOT NULL,
    [PreferowanyDS] [int] NULL,
    [OdpowiedzialnoscKarna] [bit] NOT NULL,
    [ZgodanaPrzetwDanych] [bit] NOT NULL,
    [DSPoprzednio] [bit] NOT NULL,
    [idStatus] [int] NOT NULL,
    [PrzyznanyDS] [int] NULL,
    [idStudent] [int] NOT NULL,
    [uzasadnienie] [varchar](255) NULL,
    CONSTRAINT [PK_Wniosek] PRIMARY KEY CLUSTERED
(
    [idWniosek] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
ALTER TABLE [dbo].[Wniosek] WITH CHECK ADD CONSTRAINT [FK_Wniosek_Akademik_pref] FOREIGN
KEY([PreferowanyDS])
REFERENCES [dbo].[Akademik] ([idAkademik])
GO
ALTER TABLE [dbo].[Wniosek] CHECK CONSTRAINT [FK_Wniosek_Akademik_pref]
GO
ALTER TABLE [dbo].[Wniosek] WITH CHECK ADD CONSTRAINT [FK_Wniosek_Akademik_przyzn] FOREIGN
KEY([PrzyznanyDS])
REFERENCES [dbo].[Akademik] ([idAkademik])
ON DELETE SET NULL
GO
ALTER TABLE [dbo].[Wniosek] CHECK CONSTRAINT [FK_Wniosek_Akademik_przyzn]
GO
ALTER TABLE [dbo].[Wniosek] WITH CHECK ADD CONSTRAINT [FK_Wniosek_Student] FOREIGN
KEY([idStudent])
REFERENCES [dbo].[Student] ([idStudent])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Wniosek] CHECK CONSTRAINT [FK_Wniosek_Student]
GO
ALTER TABLE [dbo].[Wniosek] WITH CHECK ADD CONSTRAINT [FK_Wniosek_WniosekStatus] FOREIGN
KEY([idStatus])
REFERENCES [dbo].[WniosekStatus] ([idStatus])
GO
ALTER TABLE [dbo].[Wniosek] CHECK CONSTRAINT [FK_Wniosek_WniosekStatus]
GO
(...)
/***** Object: StoredProcedure [dbo].[lista2]      Script Date: 01/20/2011 04:24:48 *****/
CREATE PROCEDURE [dbo].[lista2]
    -- Add the parameters for the stored procedure here
    @idWydzial int,
    @typ int
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT sw.Imie, sw.Nazwisko
    FROM StudentWniosek sw
    WHERE sw.idWydzial = @idWydzial AND sw.idStatus = @typ
END
GO
```

# Dodatek C. Fragmenty kodu źródłowego

## C1. Przyznanie miejsca

```
/// <summary>
/// reakcja na przycisk "przyznaj miejsce", dokonuje odpowiednich dla przyznania miejsca zmian
/// we wniosku
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
protected void Button1_Click(object sender, EventArgs e)
{
    //sprawdzenie czy wybrany akademik i czy są jeszcze miejsca
    if (DropDownList1.SelectedValue != "-1" && int.Parse(TextBox2.Text) > 0)
    {
        if (GridView1.SelectedValue != null)//sprawdzenie czy wybrany wniosek
        {
            ZmienStatus(4);
            //wpisanie id wybranego akademika
            int idAkademik = int.Parse(DropDownList1.SelectedValue);
            //zwiększenie ilości zajętych miejsc
            Klasy.MiejscaEtapClass.ZwiekszIloscZajetych(idWydzial, idAkademik, etap);
            WysliMail(1, idAkademik);//wysłanie informacji na maila
            PrzyznajLabel.Visible = false;
        }
        else
        {
            PrzyznajLabel.Visible = true;
        }
        GridView1.DataBind();
        TextBox1.Text = Convert.ToString(Klasy.WnioskiClass.IleWnioskow(idWydzial, 2));
    }
}
```

## C2. Sprawdzenie poprawności wypełnienia formularzu adresu

```
/// <summary>
/// Sprawdzenie czy pola są odpowiednio wypełnione
/// </summary>
/// <returns>true - jeśli pola są odpowiednio wypełnione, false - jeśli pola nie są
/// odpowiednio wypełnione</returns>
public bool SprawdzWypełnienie()
{
    bool wyslij = true;
    if (WojewodztwoDropDownList.SelectedValue == "-1")//sprawdzenie czy wybrane województwo
    {
        LabelWojewodztwo.Visible = true;//ustawienie widoczności gwiazdki z tooltip
        wyslij = false;
    }
    else
        LabelWojewodztwo.Visible = false;
    if (PowiatDropDownList.SelectedValue == "-1")//sprawdzenie czy wybrany powiat
    {
        LabelPowiat.Visible = true;//ustawienie widoczności gwiazdki z tooltip
        wyslij = false;
    }
    else
        LabelPowiat.Visible = false;
    if (GminaDropDownList.SelectedValue == "-1")//sprawdzenie czy wybrana gmina
    {
        LabelGmina.Visible = true;//ustawienie widoczności gwiazdki z tooltip
        wyslij = false;
    }
    else
        LabelGmina.Visible = false;
    if (UlicaTextBox.Text == "")//sprawdzenie czy wpisana wartość w pole tekstowe
    {
        LabelUlica.Visible = true;//ustawienie widoczności gwiazdki z tooltip
    }
}
```

```

        wyslij = false;
    }
    else
        LabelUlica.Visible = false;
    if (NrDomTextBox.Text == "")//sprawdzenie czy wpisana wartość w pole tekstowe
    {
        LabelNrDomu.Visible = true;//ustawienie widoczności gwiazdki z tooltip
        wyslij = false;
    }
    else
    {
        LabelNrDomu.Visible = false;
        int.Parse(NrDomTextBox.Text);
    }
    if (NrMieszkTextBox.Text != "")//sprawdzenie czy wpisana wartość to liczba
    {
        try
        {
            LabelNrMieszkania.Visible = false;
            int.Parse(NrMieszkTextBox.Text);
        }
        catch (Exception e)
        {
            LabelNrMieszkania.Visible = true;//ustawienie widoczności gwiazdki
            wyslij = false;
        }
    }
    return wyslij;
}

```

### C3. Dodanie nowego adresu do bazy danych

```

/// <summary>
/// Dodaje nowe elementy do tablicy Adres w bazie danych
/// </summary>
/// <returns>true - zapis się powiódł, false - zapis nie został dokonany</returns>
public bool Dodaj()
{
    SPMDDataDataContext SPMDData = PolonczenieDB.DBContextInstance;//ustawienie
połączenia z bazą
    if(this.nrBudyunku != "" && this.idGmina != 0 && this.ulica!="")
    {
        Adre adr = new Adre();//utworzenie nowego elementu bazy
        adr.idGmina = this.idGmina;
        adr.Ulica = this.ulica;
        adr.nrBudyunku = this.nrBudyunku;
        adr.nrMieszkania = this.nrMieszkania;
        //ustalenie identyfikatora adresu dla nowego elementu
        adr.idAdres = (from ad in SPMDData.Adres select ad.idAdres).Max() + 1;
        idAdres = adr.idAdres;
        SPMDData.Adres.InsertOnSubmit(adr);//zapis do bazy
        SPMDData.SubmitChanges();//potwierdzenie zapisu
    }
    else
        return false;
    return true;
}

```



# Opis zawartości płyty cd

Nieodłączną częścią niniejszej pracy jest płyta CD zawierająca:

- tekst pracy w formatach doc, odt i pdf;
- folder "SPM" zawierający kod źródłowy aplikacji,
- folder "Baza Danych" zawierający skrypty bazy danych w formacie sql,
- folder "Publish SPM" z plikami przeznaczonymi do publikacji,
- schemat bazy danych w formacie jpg,
- diagram przypadków użycia w formacie jpg.