WYŻSZA SZKOŁA BIZNESU - NATIONAL-LOUIS UNIVERSITY
W NOWYM SĄCZU

WYDZIAŁ INFORMATYKI
KIERUNEK: INFORMATYKA
SPECJALNOŚĆ: SIECI KOMPUTEROWE

Łukasz Zgłobicki
8610

SYSTEM WSPOMAGAJĄCY ZARZĄDZANIE SZPITALEM

HOSPITAL MANAGEMENT SUPPORT SYSTEM

Praca licencjacka
Promotor: dr Henryk Telega

Nowy Sącz, 2007

1. Wstęp.

1.1. Cel i zakres pracy.

Celem pracy było zaprojektowanie i stworzenie systemu wspomagającego zarządzanie szpitalem. Dowiedziałem się, że tego typu projekty cieszą się stosunkowo dużym zainteresowaniem ze strony placówek medycznych, co sprawiło, że zdecydowałem się na wybór właśnie takiego tematu pracy. System Aesculap zaprojektowałem tak, by spełniał rzeczywiste wymagania stawiane przez pracowników szpitali. Do wymagań tych można zaliczyć: przyspieszenie rejestracji pacjentów, ułatwienie planowania wizyt lekarskich i zabiegów, zwiększenie dostępności do danych historii chorób pacjentów, automatyzację procesu planowania harmonogramu dyżurów lekarskich oraz udostępnienie każdemu lekarzowi przydatnych danych dotyczących chorób i dopuszczonych do użytku leków. Ustalając listę wymagań funkcjonalnych dla projektu, zasięgnąłem opinii osób pracujących zawodowo w szpitalu. Doszedłem do wniosku, że system powinien mieć budowę modułową. Poszczególne moduły różnią się między sobą przeznaczeniem. Dwa z nich działają na platformie Windows, a jeden na platformie Windows Mobile. Aplikacja Aesculap R działająca na komputerze stacjonarnym służy do zarządzania danymi pacjentów, wizytami lekarskimi i zabiegami. Aesculap P, to aplikacja również przeznaczona na komputer stacjonarny jest narzędziem automatycznie generującym harmonogram dyżurów lekarskich i umożliwiającym jego ręczną edycję. Trzecia część systemu, Aesculap H, działa na platformie Windows Mobile. Umożliwia przeglądanie i edycję historii chorób, zapisywanie informacji na temat stanu zdrowia pacjentów, korzystanie z uaktualnianych informacji o chorobach i lekach oraz przeglądanie harmonogramu dyżurów lekarskich. Te trzy uzupełniające się moduły przyspieszają pracę z danymi różnego typu.

1.2. Zawartość poszczególnych rozdziałów.

W dalszej części rozdziału 1 wymienione zostały wszystkie funkcje poszczególnych modułów systemu Aesculap oraz technologie użyte w procesie tworzenia systemu i uzasadniony został ich wybór. W rozdziale tym został także zawarty zwięzły opis prac projektowych i implementacyjnych. Rozdział 2 zawiera informacje na temat bazy danych *aesculap2* – diagram tabel i relacji pomiędzy nimi, a także wykaz indeksów unikalnych, wyzwalaczy i zapytań. W rozdziale 3 omówiona została struktura systemu Aesculap. Rozdział ten zawiera wykaz wszystkich klas poszczególnych modułów systemu i kod źródłowy wybranych klas. Rozdział 4 stanowi przewodnik użytkownika z dołączonymi zrzutami ekranu. W rozdziale 5 przedstawiona została obsługa błędów użytkownika. Rozdział 6 określa wymagania sprzętowe i programowe, które muszą zostać spełnione aby uruchomić moduły systemu Aesculap i traktuje o instalacji poszczególnych modułów . Rozdział 7 stanowi podsumowanie i zawiera wnioski dotyczące pracy nad projektem. Rozdział 8 zawiera listę źródeł informacji, z których korzystałem tworząc projekt.

1.3. Wykaz wszystkich funkcji poszczególnych modułów systemu, który powstał w ramach pracy.

Funkcje modułu Aesculap R:

- Rejestracja pacjentów.
- Edycja danych zarejestrowanych pacjentów.
- Rejestracja przyjęć pacjentów.
- Rejestracja wypisów pacjentów.
- Planowanie wizyt lekarskich.
- Edycja danych zaplanowanych wizyt lekarskich.
- Anulowanie zaplanowanych wizyt lekarskich.
- Przeglądanie zaplanowanych wizyt lekarskich.
- Planowanie zabiegów.
- Edycja danych zaplanowanych zabiegów.
- Anulowanie zaplanowanych zabiegów.
- Przeglądanie zaplanowanych zabiegów.
- Wyszukiwanie pacjentów według numeru PESEL.
- Wyszukiwanie pacjentów według nazwiska.

Funkcje modułu Aesculap P:

- Planowanie hamonogramu dyżurów lekarskich.
- Edycja harmonogramu dyżurów lekarskich.
- Przeglądanie harmonogramu dyżurów lekarskich.
- Edycja danych o urlopach lekarzy.
- Edycja danych o godzinach pracy lekarzy.
- Drukowanie danych dotyczących przebiegu procesu planowania.

Funkcje modułu Aesculap H:

- Przeglądanie zaplanowanych wizyt lekarskich.

- Przeglądanie zaplanowanych zabiegów.
- Edycja czynności do wykonania dla każdego zaplanowanego zabiegu.
- Przeglądanie przyjęć pacjentów.
- Edycja rozpoznań powiązanych z przyjęciami pacjentów.
- Rejestracja zgonów pacjentów.
- Przeglądanie dostępnych leków.
- Przeglądanie informacji o chorobach.
- Edycja obserwacji stanu zdrowia pacjentów.
- Edycja historii chorób pacjentów.
- Przeglądanie historii chorób pacjentów.
- Edycja historii użytych leków.
- Przeglądanie historii użytych leków.
- Wyszukiwanie zaplanowanych wizyt według daty.
- Wyszukiwanie zaplanowanych zabiegów według daty.
- Wyszukiwanie czynności do wykonania podczas zabiegu według numeru zabiegu.
- Wyszukiwanie zarejestrowanych przyjęć pacjentów w określonym przedziale czasu.
- Wyszukiwanie zarejestrowanych zgonów pacjentów według daty.
- Wyszukiwanie leków według producenta.
- Wyszukiwanie leków według nazwy leku.
- Wyszukiwanie zdarzeń historii chorób według daty.
- Wyszukiwanie zaplanowanych dyżurów lekarskich według daty.
- Wyszukiwanie zaplanowanych dyżurów lekarskich według identyfikatora lekarza.

1.4. Użyte technologie i uzasadnienie ich wyboru.

Technologie użyte podczas tworzenia systemu, to:

- **C#**
- **Microsoft Visual Studio 2005**
- **Microsoft SQL Server 2005**
- **Microsoft SQL Server 2005 Compact Edition**

C#

Wybór języka programowania C# podyktowany był tym, że został on zaprojektowany właśnie dla firmy Microsoft i przy jego użyciu można tworzyć aplikacje przeznaczone na platformę Windows Mobile. Zastosowanie tego samego języka podczas projektowania wszystkich modułów systemu, a także jego pokrewieństwo z językami C++ i Java w pewnym stopniu ułatwiło ten proces.

Microsoft Visual Studio 2005

Biorąc pod uwagę konieczność przygotowania modułu działającego na platformie Windows Mobile, Microsoft Visual Studio 2005 wydaje się być jedynym słusznym wyborem. W porównaniu do Microsoft Visual Studio .NET 2003, oferuje większą liczbę emulatorów z większymi możliwościami. Testy aplikacji są bardzo dokładne i kompletne, gdyż emulatory operują na identycznych instrukcjach jak komputer kieszonkowy, posiadają ten sam model pamięci i używają tych samych sterowników, co prawdziwe urządzenie. Zaletą jest też korzystanie z tych samych skompilowanych plików jako urządzenie, a zatem nie jest konieczna kompilacja oddzielnie dla emulatora i urządzenia.

Microsoft SQL Server 2005

Konsekwencją decyzji o użyciu Microsoft Visual Studio 2005 jest wybór relacyjnego systemu zarządzania bazami danych Microsoft SQL Server 2005. Microsoft Visual Studio 2005 zapewnia bezpośrednie połączenie

ze wspomnianą bazą, co znacznie ułatwia pracę z nią. Ponadto konfiguracja replikacji pomiędzy bazą Microsoft SQL Server 2005 i Microsoft SQL Server Compact Edition nie przysparza problemów.

Microsoft SQL Server 2005 Compact Edition

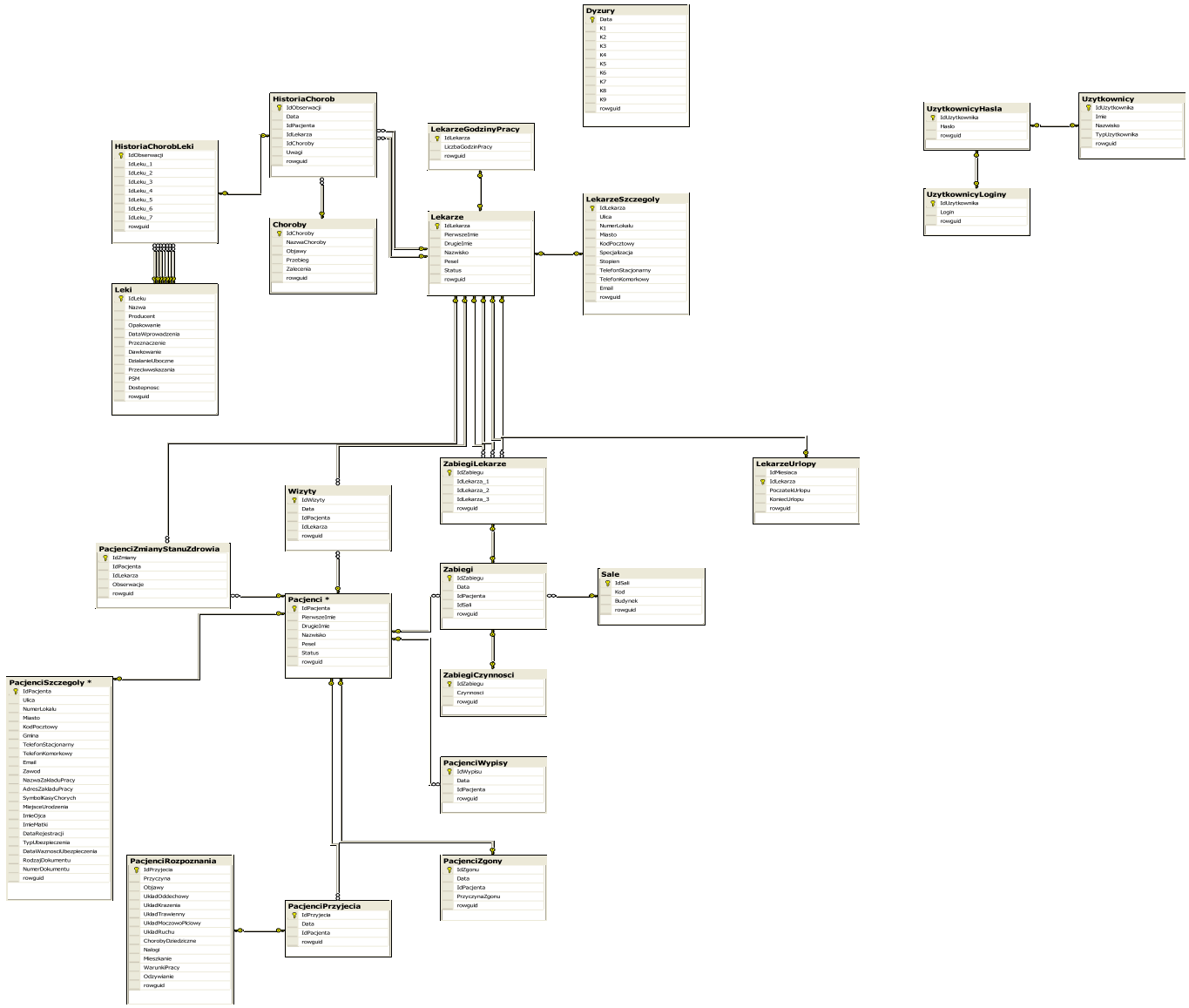
Wybór systemu Microsoft SQL Server 2005 Compact Edition dla urządzenia mobilnego jest ściśle związany z decyzją o wyborze serwera stacjonarnego Microsoft SQL Server 2005. Pozwala na konfigurację replikacji typu Merge niezbędnej dla synchronizacji danych pomiędzy urządzeniem mobilnym i urządzeniem stacjonarnym. Synchronizacja ta stanowi jedno z wymagań funkcjonalnych systemu.

1.5. Zwięzły opis prac projektowych i implementacyjnych.

Kod źródłowy systemu Aesculap zawiera ponad 75000 linii. Utworzona na potrzeby projektu baza danych zawiera 24 tabele. Zaprojektowanie bazy danych i zabezpieczenie jej przed niepożądanym zachowaniem użytkowników poprzez użycie indeksów unikalnych i wyzwalaczy zajęło stosunkowo dużo czasu. Podobnie było z projektowaniem algorytmu do planowania grafiku dyżurów lekarskich. Prace nad nim odbywały się w kilku etapach. Pierwszym projektowanym modułem był Aesculap H, jednak po rozwiązaniu drobnych problemów z replikacją, prace nad nim zostały chwilowo przerwane na rzecz modułu Aesculap R. Następne w kolejności były: moduł Aesculap P i doprowadzenie modułu Aesculap H do ostatecznej postaci. Wprowadzane zmiany dotyczyły głównie modułów Aesculap R i Aesculap H, gdyż były związane z uzyskiwanymi na bieżąco informacjami od pracowników placówek medycznych. W rezultacie powstał sporych rozmiarów projekt, który może być z powodzeniem rozwijany i dopasowywany do potrzeb użytkowników.

2. Baza danych.

2.1. Struktura bazy danych.



Rys. 1. Tabele bazy danych aesculap2 i relacje pomiędzy nimi.

2.2. Indeksy unikalne.

Konieczność użycia danych unikalnych dla każdego pacjenta, lekarza i użytkownika systemu poskutkowała utworzeniem przedstawionych poniżej indeksów unikalnych.

Indeks unikalny utworzony dla pól Pesel w tabeli Pacjenci. Każdy pacjent posiada unikalny numer PESEL.

```
CREATE UNIQUE INDEX UixPeselPacjenci ON Pacjenci(Pesel)
```

Indeks unikalny utworzony dla pól Pesel w tabeli Lekarze. Każdy lekarz posiada swój numer PESEL.

```
CREATE UNIQUE INDEX UixPeselLekarze ON Lekarze(Pesel)
```

Indeks unikalny utworzony dla pól Email w tabeli Lekarze. Każdy lekarz posiada unikalny Email.

```
CREATE UNIQUE INDEX UixEmailLekarze ON LekarzeSzczegoly(Email)
```

Indeks unikalny utworzony dla pól NazwaChoroby w tabeli Choroby. Każda choroba posiada unikalną nazwę.

```
CREATE UNIQUE INDEX UixNazwaChoroby ON Choroby(NazwaChoroby)
```

Indeks unikalny utworzony dla pól NazwaLeku w tabeli Leki. Każdy lek posiada unikalną nazwę.

```
CREATE UNIQUE INDEX UixNazwaLeku ON Leki(NazwaLeku)
```

Indeks unikalny utworzony dla pól NumerDokumentu w tabeli PacjenciSzczegoly. Każdy dokument ubezpieczenia posiada unikalny numer.

```
CREATE UNIQUE INDEX UixNumerDokumentu ON PacjenciSzczegoly(NumerDokumentu)
```

Indeks unikalny utworzony dla pól IdPacjenta w tabeli PacjenciZgony. Każdy akt zgonu posiada unikalny identyfikator pacjenta.

```
CREATE UNIQUE INDEX UixZgonyIdPacjenta ON PacjenciZgony(IdPacjenta)
```

Indeks unikalny utworzony dla pól Kod w tabeli Sale. Każda sala posiada unikalny kod.

```
CREATE UNIQUE INDEX UixSaleKod ON Sale(Kod)
```

Indeks unikalny utworzony dla pól Login w tabeli UzytkownicyLoginy. Każdy użytkownik posiada unikalny login.

```
CREATE UNIQUE INDEX UixUzytkownicyLogin ON UzytkownicyLoginy(Login)
```

2.3. Zapytania do bazy danych.

Poniżej zostały przedstawione zapytania wypełniające poszczególne tabele *aesculap2DataSet*.

Zapytanie wypełniające tabelę *Lekarze*.

```
SELECT IdLekarza, PierwszeImie, DrugieImie, Nazwisko, Pesel, Status FROM
dbo.Lekarze
```

Zapytanie wypełniające tabelę *LekarzeSzczegoly*.

```
SELECT IdLekarza, Ulica, NumerLokalu, Miasto, KodPocztowy, Specjalizacja,
Stopien, TelefonStacjonarny, TelefonKomorkowy, Email FROM dbo.LekarzeSzczegoly
```

Zapytanie wypełniające tabelę *LekarzeGodzinyPracy*.

```
SELECT IdLekarza, LiczbaGodzinPracy FROM dbo.LekarzeGodzinyPracy
```

Zapytanie wypełniające tabelę *LekarzeUrlopy*.

```
SELECT IdMiesiaca, IdLekarza, PoczatekUrlopu, KoniecUrlopu FROM
dbo.LekarzeUrlopy
```

Zapytanie wypełniające tabelę *Pacjenci*.

```
SELECT IdPacjenta, PierwszeImie, DrugieImie, Nazwisko, Pesel, Status FROM
dbo.Pacjenci
```

Zapytanie wypełniające tabelę *PacjenciSzczegoly*.

```
SELECT IdPacjenta, Ulica, NumerLokalu, Miasto, KodPocztowy, Gmina,
TelefonStacjonarny, TelefonKomorkowy, Email, Zawod, NazwaZakladuPracy,
AdresZakladuPracy, SymbolKasyChorych, MiejsceUrodzenia, ImieOjca, ImieMatki,
DataRejestracji, TypUbezpieczenia, DataWaznosciUbezpieczenia, RodzajDokumentu,
NumerDokumentu FROM dbo.PacjenciSzczegoly
```

Zapytanie wypełniające tabelę *PacjenciPrzyjecia*.

```
SELECT IdPrzyjecia, Data, IdPacjenta FROM dbo.PacjenciPrzyjecia
```

Zapytanie wypełniające tabelę *PacjenciWypisy*.

```
SELECT IdWypisu, Data, IdPacjenta FROM dbo.PacjenciWypisy
```

Zapytanie wypełniające tabelę *PacjenciZgony*.

```
SELECT IdZgonu, Data, IdPacjenta, PrzyczynaZgonu FROM dbo.PacjenciZgony
```

Zapytanie wypełniające tabelę *PacjenciZmianyStanuZdrowia*.

```
SELECT IdZmiany, IdPacjenta, IdLekarza, Obserwacje FROM
dbo.PacjenciZmianyStanuZdrowia
```

Zapytanie wypełniające tabelę *Wizyty*.

```
SELECT IdWizyty, Data, IdPacjenta, IdLekarza FROM dbo.Wizyty
```

Zapytanie wypełniające tabelę *Zabiegi*.

```
SELECT IdZabiegu, Data, IdPacjenta, IdSali FROM dbo.Zabiegi
```

Zapytanie wypełniające tabelę *ZabiegiLekarze*.

```
SELECT IdZabiegu, IdLekarza_1, IdLekarza_2, IdLekarza_3 FROM dbo.ZabiegiLekarze
```

Zapytanie wypełniające tabelę *ZabiegiCzynnosci*.

```
SELECT IdZabiegu, Czynnosci FROM dbo.ZabiegiCzynnosci
```

Zapytanie wypełniające tabelę *Sale*.

```
SELECT IdSali, Kod, Budynek FROM dbo.Sale
```

Zapytanie wypełniające tabelę *HistoriaChorob*.

```
SELECT IdObserwacji, Data, IdPacjenta, IdLekarza, IdChoroby, Uwagi FROM
dbo.HistoriaChorob
```

Zapytanie wypełniające tabelę *HistoriaChorobLeki*.

```
SELECT IdObserwacji, IdLeku_1, IdLeku_2, IdLeku_3, IdLeku_4, IdLeku_5,
IdLeku_6, IdLeku_7 FROM dbo.HistoriaChorobLeki
```

Zapytanie wypełniające tabelę *Leki*.

```
SELECT IdLeku, Nazwa, Producent, Opakowanie, DataWprowadzenia, Przeznaczenie,
Dawkowanie, DzialanieUboczne, Przeciwwskazania, PSM, Dostepnosc FROM dbo.Leki
```

Zapytanie wypełniające tabelę *Choroby*.

```
SELECT IdChoroby, NazwaChoroby, Objawy, Przebieg, Zalecenia FROM dbo.Choroby
```

Zapytanie wypełniające tabelę *Dyzury*.

```
SELECT Data, Lekarz_1, Lekarz_2, Lekarz_3, Lekarz_4, Lekarz_5, Lekarz_6,
LekarzDodatkowy_1, LekarzDodatkowy_2, LekarzDodatkowy_3 FROM dbo.Dyzury
```

Zapytanie wypełniające tabelę *Uzytkownicy*.

```
SELECT IdUzytkownika, Imie, Nazwisko, TypUzytkownika FROM dbo.Uzytkownicy
```

Zapytanie wypełniające tabelę *UzytkownicyLoginy*.

```
SELECT IdUzytkownika, Login FROM dbo.UzytkownicyLoginy
```

Zapytanie wypełniające tabelę *UzytkownicyHasla*.

```
SELECT IdUzytkownika, Haslo FROM dbo.UzytkownicyHasla
```

Poniżej zostały przedstawione zapytania używane do wypełniania poszczególnych tabel *aesculap2DataSet* z wykorzystaniem parametrów podanych przez użytkownika.

Zapytanie wypełniające tabelę *Pacjenci* na podstawie podanego numeru PESEL (Aesculap R).

```
SELECT     IdPacjenta, PierwszeImie, DrugieImie, Nazwisko, Pesel, Status
FROM         Pacjenci
WHERE     (Pesel = @Pesel)
```

Zapytanie wypełniające tabelę *Pacjenci* na podstawie podanego nazwiska (Aesculap R).

```
SELECT     IdPacjenta, PierwszeImie, DrugieImie, Nazwisko, Pesel, Status
FROM         Pacjenci
WHERE     (Nazwisko = @Surname)
```

Zapytanie wypełniające tabelę *HistoriaChorob* na podstawie podanej daty (Aesculap H).

```
SELECT     IdObserwacji, Data, IdPacjenta, IdLekarza, IdChoroby, Uwagi, rowguid
FROM         HistoriaChorob
WHERE     (Data = @Data)
```

Zapytanie wypełniające tabelę *Leki* na podstawie podanej nazwy producenta (Aeculap H).

```
SELECT     IdLeku, Nazwa, Producent, Opakowanie, DataWprowadzenia,
Przeznaczenie, Dawkowanie, DzialanieUboczne, Przeciwwskazania, PSM, Dostepnosc,
       rowguid
FROM         Leki
WHERE     (Producent = @Manufacturer)
```

Zapytanie wypełniające tabelę *Leki* na podstawie podanej nazwy leku (Aesculap H).

```
SELECT     IdLeku, Nazwa, Producent, Opakowanie, DataWprowadzenia,
Przeznaczenie, Dawkowanie, DzialanieUboczne, Przeciwwskazania, PSM, Dostepnosc,
       rowguid
FROM         Leki
WHERE     (Nazwa = @Nazwa)
```

Zapytanie wypełniające tabelę *PacjeciPrzyjecia* na podstawie podanej daty początkowej i końcowej (Aesculap H).

```
SELECT     IdPrzyjecia, Data, IdPacjenta, rowguid
FROM         PacjenciPrzyjecia
WHERE     (Data BETWEEN @Data1 AND @Data2)
```

Zapytanie wypełnające tabelę *PacjenciZgony* na podstawie podanej daty (Aesculap H).

```
SELECT     IdZgonu, Data, IdPacjenta, PrzyczynaZgonu, rowguid
FROM         PacjenciZgony
WHERE     (Data = @Date)
```

Zapytanie wypełniające tabelę *Wizyty* na podstawie podanej daty (Aesculap H).

```
SELECT     IdWizyty, Data, IdPacjenta, IdLekarza, rowguid
FROM         Wizyty
WHERE     (Data >= @Date)
```

Zapytanie wypełniające tabelę *Dyzury* na podstawie podanych identyfikatorów lekarzy (Aesculap H).

```
SELECT     Data, K1, K2, K3, K4, K5, K6, K7, K8, K9, rowguid
FROM         Dyzury
WHERE     (K1 = @Id1) OR
       (K2 = @Id2) OR
       (K3 = @Id3) OR
       (K4 = @Id4) OR
       (K5 = @Id5) OR
       (K6 = @Id6) OR
       (K7 = @Id7)
```

Zapytanie wypełniające tabelę *Dyzury* na podstawie podanej daty (Aesculap H).

```
SELECT      Data, K1, K2, K3, K4, K5, K6, K7, K8, K9, rowguid
FROM          Dyzury
WHERE      (Data = @Data)
```

2.4. Wyzwalacze.

Prawdopodobieństwo popełnienia przez użytkownika błędu podczas wprowadzania danych poskutkowało utworzeniem przedstawionego poniżej zestawu wyzwalaczy, które skutecznie zapobiegają błędnemu wprowadzaniu lub modyfikacji danych.

Wyzwalacz zapobiegający dopisaniu do listy lekarzy wykonujących zabieg lekarza, który posiada status *NZ*, czyli nie pracuje przez okres dłuższy niż 1 dzień lub lekarza, który nie pracuje w danym dniu.

```
CREATE TRIGGER [dbo].[ZabiegiLekarzePrev]
ON [dbo].[ZabiegiLekarze]
FOR INSERT, UPDATE
AS
IF UPDATE (IdLekarza_1) OR UPDATE (IdLekarza_2) OR UPDATE (IdLekarza_3)

IF EXISTS (SELECT * FROM ZabiegiLekarze zl, Lekarze l, Inserted i WHERE
zl.IdLekarza_1 = i.IdLekarza_1 AND l.IdLekarza = zl.IdLekarza_1 AND l.Status =
'NZ')
OR
 EXISTS (SELECT * FROM ZabiegiLekarze zl, Lekarze l, Inserted i WHERE
zl.IdLekarza_2 = i.IdLekarza_2 AND l.IdLekarza = zl.IdLekarza_2 AND l.Status =
'NZ')
OR
 EXISTS (SELECT * FROM ZabiegiLekarze zl, Lekarze l, Inserted i WHERE
zl.IdLekarza_3 = i.IdLekarza_3 AND l.IdLekarza = zl.IdLekarza_3 AND l.Status =
'NZ')
OR
 NOT EXISTS (SELECT * FROM Dyzury d, Inserted i, Zabiegi z WHERE z.IdZabiegu =
i.IdZabiegu AND d.Data = z.Data AND (d.K1 = i.IdLekarza_1 OR d.K2 =
i.IdLekarza_1 OR d.K3 = i.IdLekarza_1 OR d.K4 = i.IdLekarza_1 OR d.K5 =
i.IdLekarza_1 OR d.K6 = i.IdLekarza_1 OR d.K7 = i.IdLekarza_1 OR d.K8 =
i.IdLekarza_1 OR d.K9 = i.IdLekarza_1))
OR
 NOT EXISTS (SELECT * FROM Dyzury d, Inserted i, Zabiegi z WHERE z.IdZabiegu =
i.IdZabiegu AND d.Data = z.Data AND (d.K1 = i.IdLekarza_2 OR d.K2 =
i.IdLekarza_2 OR d.K3 = i.IdLekarza_2 OR d.K4 = i.IdLekarza_2 OR d.K5 =
i.IdLekarza_2 OR d.K6 = i.IdLekarza_2 OR d.K7 = i.IdLekarza_2 OR d.K8 =
i.IdLekarza_2 OR d.K9 = i.IdLekarza_2))
OR
 NOT EXISTS (SELECT * FROM Dyzury d, Inserted i, Zabiegi z WHERE z.IdZabiegu =
i.IdZabiegu AND d.Data = z.Data AND (d.K1 = i.IdLekarza_3 OR d.K2 =
i.IdLekarza_3 OR d.K3 = i.IdLekarza_3 OR d.K4 = i.IdLekarza_3 OR d.K5 =
i.IdLekarza_3 OR d.K6 = i.IdLekarza_3 OR d.K7 = i.IdLekarza_3 OR d.K8 =
i.IdLekarza_3 OR d.K9 = i.IdLekarza_3))
BEGIN
        RAISERROR ('Ten lekarz nie pracuje w tym dniu.', 16, 1)
        ROLLBACK TRAN
        RETURN
END
```

Wyzwalacz zapobiegający ustaleniu wizyty lekarskiej u lekarza, który posiada status *NZ*, czyli nie pracuje przez okres dłuższy niż 1 dzień lub lekarza, który nie pracuje w danym dniu.

```
CREATE TRIGGER [dbo].[WizytyPrev]
ON [dbo].[Wizyty]
FOR INSERT, UPDATE
AS
IF UPDATE (IdLekarza)

IF EXISTS (SELECT * FROM Wizyty w, Lekarze l, Inserted i WHERE w.IdLekarza =
i.IdLekarza AND l.IdLekarza = w.IdLekarza AND l.Status = 'NZ')


BEGIN
        RAISERROR ('Ten lekarz nie pracuje w ogole.', 16, 1)
        ROLLBACK TRAN
        RETURN
END



ELSE
IF NOT EXISTS (SELECT * FROM Dyzury d, Inserted i, Wizyty w WHERE w.Data =
i.Data AND d.Data = w.Data AND (d.K1 = i.IdLekarza OR d.K2 = i.IdLekarza OR
d.K3 = i.IdLekarza OR d.K4 = i.IdLekarza OR d.K5 = i.IdLekarza OR d.K6 =
i.IdLekarza OR d.K7 = i.IdLekarza OR d.K8 = i.IdLekarza OR d.K9 = i.IdLekarza))

BEGIN
        RAISERROR ('Ten lekarz nie pracuje w tym dniu.', 16, 1)
        ROLLBACK TRAN
        RETURN
END
```

Wyzwalacz zapobiegający popełnieniu błędu polegającego na rejestracji przyjęcia do szpitala pacjenta, który jeszcze nie został z niego wypisany.

```
CREATE TRIGGER [dbo].[PacjenciPrzyjeciaPrev]
ON [dbo].[PacjenciPrzyjecia]
FOR INSERT, UPDATE
AS
IF UPDATE (IdPacjenta) AND UPDATE (Data) AND UPDATE (IdPrzyjecia)

IF EXISTS (SELECT * FROM Inserted i JOIN PacjenciWypisy pw ON i.IdPacjenta =
pw.IdPacjenta AND i.Data < (SELECT TOP 1 Data FROM PacjenciWypisy ORDER BY Data
DESC))

BEGIN
        RAISERROR ('Nie mo¿na przyj¹æ pacjenta, który w dalszym ci¹gu jest w
szpitalu.', 16, 1)
        ROLLBACK TRAN
        RETURN
END
```

Wyzwalacz sprawdzający ilość przyjęć i wypisów pacjenta i zapobiegający rejestracji większej liczby wypisów pacjenta ze szpitala niż liczba przyjęć pacjenta do szpitala.

```
CREATE TRIGGER [dbo].[PacjenciWypisyPrev]
ON [dbo].[PacjenciWypisy]
FOR INSERT, UPDATE
```

```
AS
IF UPDATE (IdPacjenta) AND UPDATE (Data) AND UPDATE (IdWypisu)

IF (SELECT COUNT (*) FROM PacjenciWypisy pw JOIN Inserted i ON pw.IdPacjenta =
i.IdPacjenta) != (SELECT COUNT (*) FROM PacjenciPrzyjecia pp JOIN Inserted i ON
pp.IdPacjenta = i.IdPacjenta)

BEGIN
        RAISERROR ('Liczba przyjêæ i wypisów dla tego pacjenta siê nie
zgadza.', 16, 1)
        ROLLBACK TRAN
        RETURN
END
```

Wyzwalacz zapobiegający zapisowi zduplikowanych danych w tabeli *PacjenciPrzyjecia*.

```
CREATE TRIGGER [dbo].[PacjenciPrzyjeciaInsertUpdate]
ON [dbo].[PacjenciPrzyjecia]
FOR INSERT, UPDATE
AS
IF UPDATE (IdPacjenta) AND UPDATE (Data)
IF (SELECT COUNT (*) FROM PacjenciPrzyjecia pp INNER JOIN Inserted i ON
pp.IdPacjenta LIKE i.IdPacjenta AND pp.Data LIKE i.Data) > 1
BEGIN
        RAISERROR ('Ten pacjent zosta³ przyjêty o podanej godzinie.', 16, 1)
        ROLLBACK TRAN
        RETURN
END
```

Wyzwalacz zmieniający status pacjenta na *Przyjęty* po rejestracji przyjęcia go do szpitala.

```
CREATE TRIGGER [dbo].[PacjenciPrzyjeciaNewRec]
ON [dbo].[PacjenciPrzyjecia]
FOR INSERT
AS
UPDATE Pacjenci
SET Status = 'Przyjêty' WHERE Pacjenci.IdPacjenta LIKE (SELECT IdPacjenta FROM
Inserted)
```

Wyzwalacz zmieniający status pacjenta na *Nieprzyjęty* po rejestracji wypisu ze szpitala.

```
GO
CREATE TRIGGER [dbo].[PacjenciWypisyNewRec]
ON [dbo].[PacjenciWypisy]
FOR INSERT
AS
UPDATE Pacjenci
SET Status = 'Nieprzyjêty' WHERE Pacjenci.IdPacjenta LIKE (SELECT IdPacjenta
FROM Inserted)
```

Wyzwalacz zmieniający status pacjenta na *Nie żyje* po rejestracji zgonu pacjenta.

```
CREATE TRIGGER [dbo].[PacjenciZgonyNewRec]
ON [dbo].[PacjenciZgony]
FOR INSERT
AS
UPDATE Pacjenci
```

```
SET Status = 'Nie ¿yje' WHERE Pacjenci.IdPacjenta LIKE (SELECT IdPacjenta FROM
Inserted)
```

Wyzwalacz zapobiegający popełnieniu następujących błędów:

- Zapisanie po raz kolejny tego samego lekarza i tego samego pacjenta na wizytę lekarską o godzinie, na którą zostali już zapisani.
- Zapisanie po raz kolejny tego samego pacjenta na wizytę lekarską o godzinie, na którą został już zapisany.
- Zapisanie po raz kolejny tego samego lekarza na wizytę lekarską o godzinie, na którą został już zapisany.

```
CREATE TRIGGER [dbo].[WizytyInsertUpdate]
ON [dbo].[Wizyty]
FOR INSERT, UPDATE
AS
DECLARE @errormessage nvarchar(MAX)
DECLARE @lekarzImie nvarchar(MAX)
DECLARE @lekarzNazwisko nvarchar(MAX)
IF UPDATE (IdPacjenta) AND UPDATE (IdLekarza) AND UPDATE (Data)
IF (SELECT COUNT (*) FROM Wizyty w INNER JOIN Inserted i ON w.IdPacjenta LIKE
i.IdPacjenta AND w.IdLekarza LIKE i.IdLekarza

AND w.Data LIKE i.Data) > 1
BEGIN
                SELECT @lekarzImie=(SELECT PierwszeImie FROM Lekarze l INNER
JOIN Inserted i ON l.IdLekarza LIKE i.IdLekarza)
                SELECT @lekarzNazwisko=(SELECT Nazwisko FROM Lekarze l INNER
JOIN Inserted i ON l.IdLekarza LIKE i.IdLekarza)
        SELECT @errormessage = 'Ten pacjent jest ju¿ umówiony z lekarzem: ' +
@lekarzImie + ' ' + @lekarzNazwisko + ' na wizytê o tej godzinie.'
                RAISERROR ( @errormessage, 16, 1)
        ROLLBACK TRAN
        RETURN
END

ELSE

IF (SELECT COUNT (*) FROM Wizyty w INNER JOIN Inserted i ON w.IdPacjenta LIKE
i.IdPacjenta AND w.Data LIKE i.Data) > 1
BEGIN
                RAISERROR ( 'Ten pacjent jest ju¿ umówiony z lekarzem na t¹
godzinê.', 16, 1)
        ROLLBACK TRAN
        RETURN
END

ELSE

IF (SELECT COUNT (*) FROM Wizyty w INNER JOIN Inserted i ON w.Data LIKE i.Data
AND w.IdLekarza LIKE i.IdLekarza) > 1
BEGIN
                RAISERROR ( 'Ten lekarz jest ju¿ umówiony na wizytê o tej
godzinie.', 16, 1)
        ROLLBACK TRAN
        RETURN
END
```

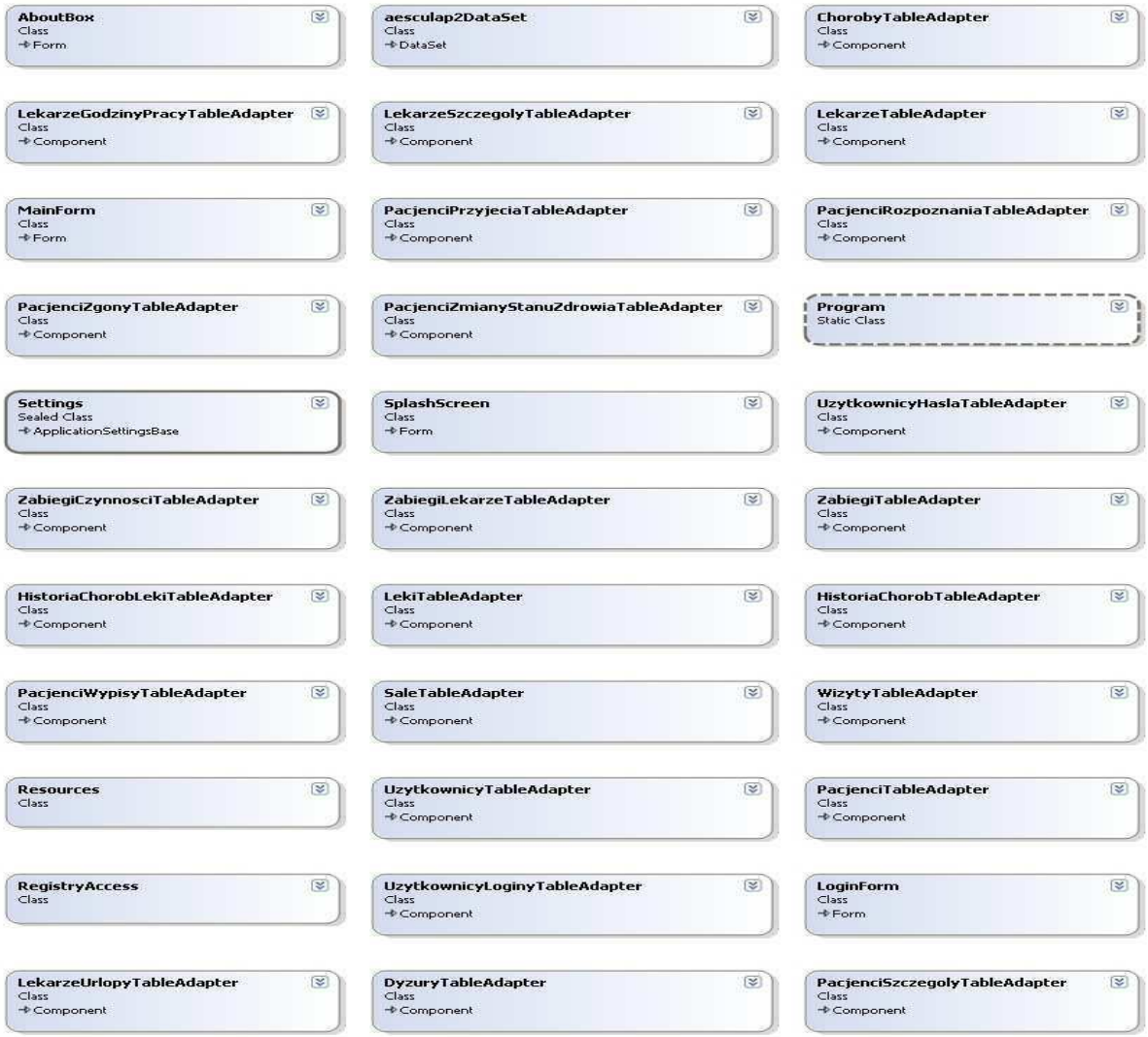
Wyzwalacz zapobiegający zapisaniu pacjenta na dwa zabiegi w ciągu jednego dnia i uniemożliwiający wybór dla zabiegu sali operacyjnej, która jest zajęta o określonej godzinie.

```
CREATE TRIGGER [dbo].[ZabiegiInsertUpdate]
ON [dbo].[Zabiegi]
FOR INSERT, UPDATE
AS
IF UPDATE (IdPacjenta) AND UPDATE (IdSali) AND UPDATE (Data)
IF (SELECT COUNT (*) FROM Zabiegi w INNER JOIN Inserted i ON w.IdPacjenta LIKE
i.IdPacjenta AND w.IdSali LIKE i.IdSali AND w.Data LIKE i.Data) > 1
BEGIN
                RAISERROR ( 'Ten pacjent ma ju¿ ustalony zabieg na ten dzieñ w
tej sali.', 16, 1)
        ROLLBACK TRAN
        RETURN
END

ELSE

IF (SELECT COUNT (*) FROM Zabiegi w INNER JOIN Inserted i ON w.IdPacjenta LIKE
i.IdPacjenta AND w.Data LIKE i.Data) > 1
BEGIN
                RAISERROR ( 'Ten pacjent ma ju¿ ustalony zabieg na ten dzieñ.',
16, 1)
        ROLLBACK TRAN
        RETURN
END
```

3. Struktura systemu.

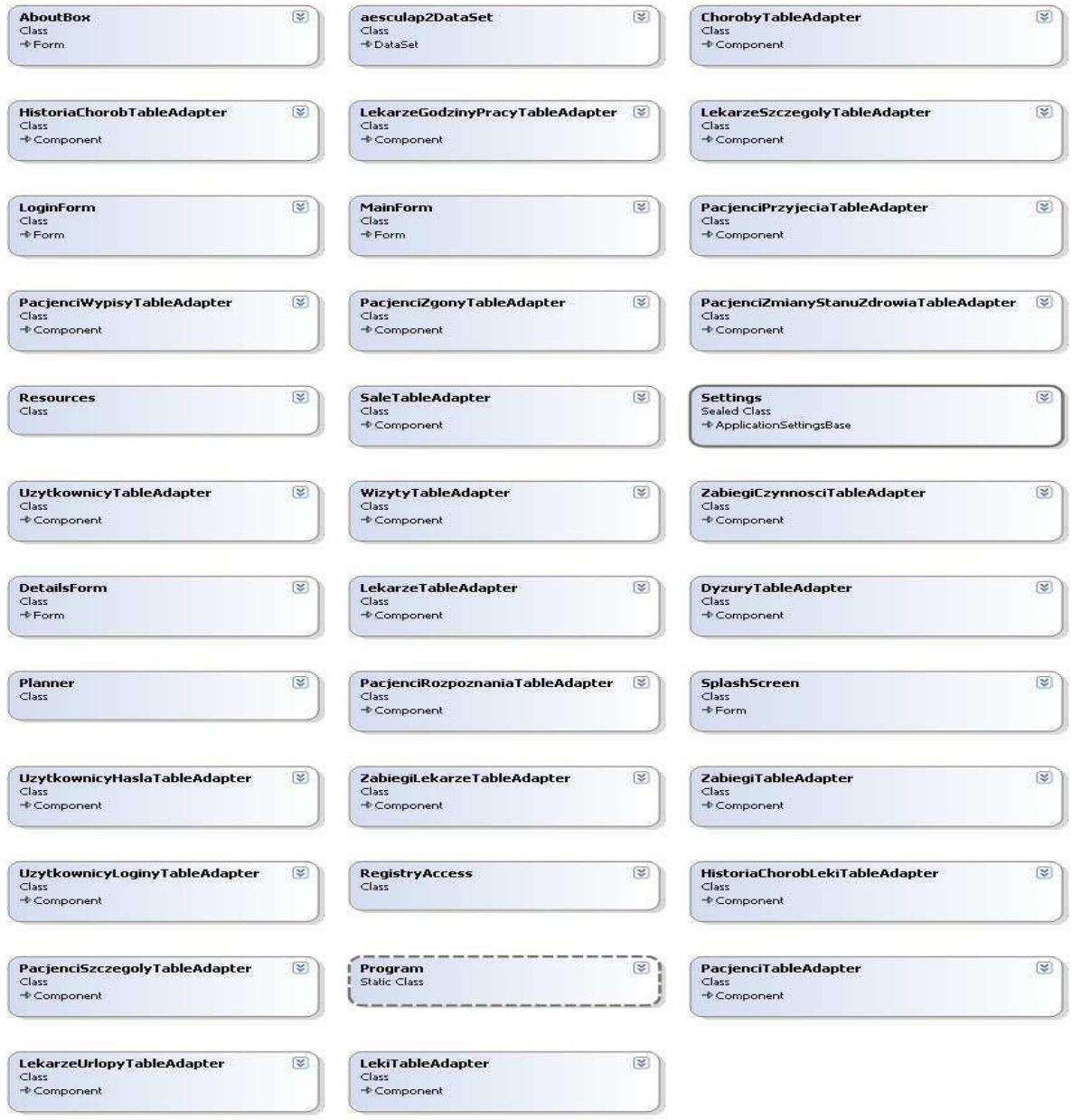
3.1. Wykaz wszystkich klas poszczególnych modułów.

- Klasy modułu Aesculap R



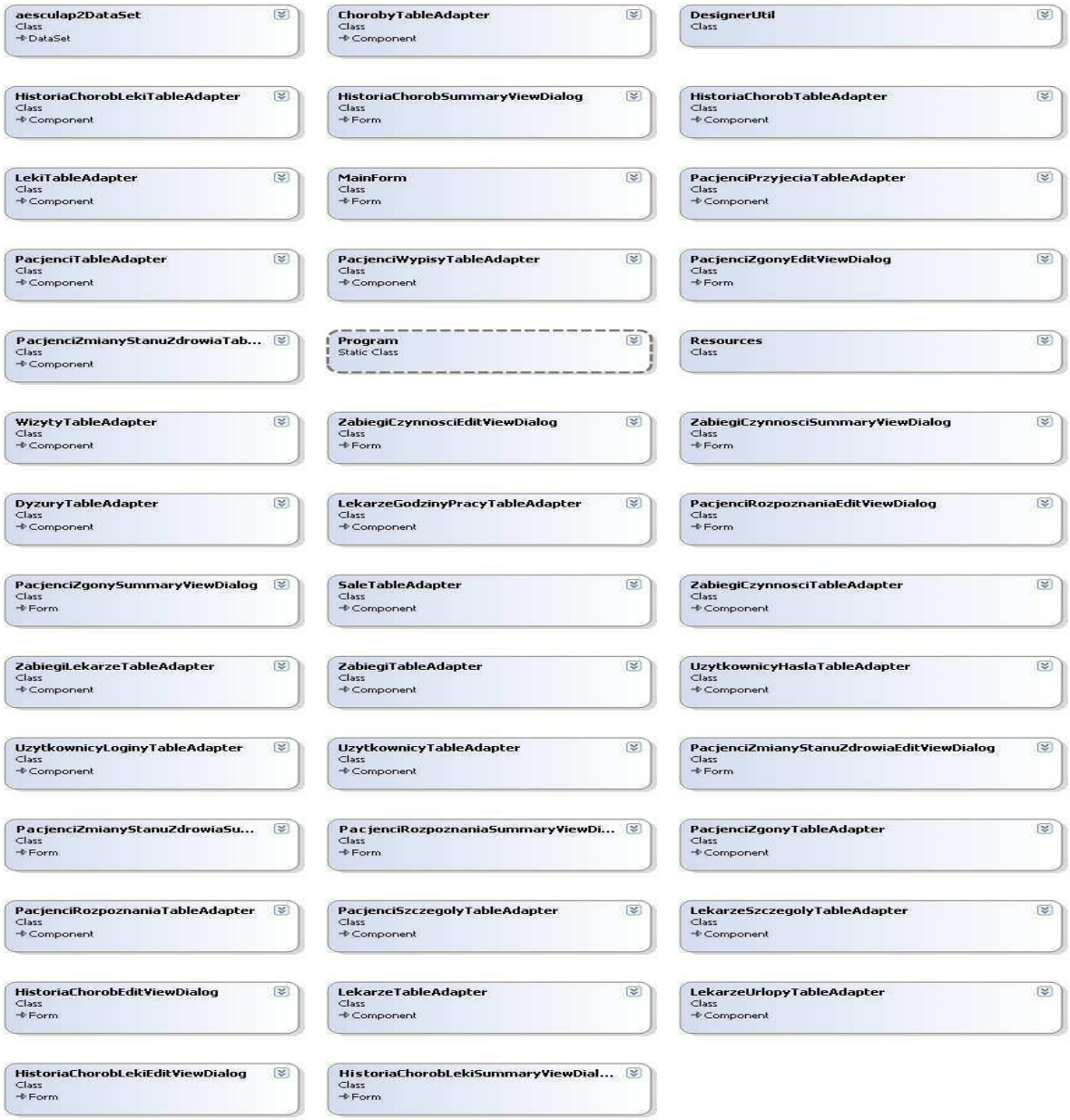
Rys. 2. Klasy modułu Aesculap R.

- Klasy modułu Aesculap P



Rys. 3. Klasy modułu Aesculap P.

- Klasy modułu Aesculap H



Rys. 4. Klasy modułu Aesculap H.

3.2. Kod źródłowy wybranych klas.

Cały kod źródłowy systemu Aesculap dostępny jest na dołączonej płycie DVD. Ze względu na duży rozmiar projektu, w tej części rozdziału zamieszczony został kod źródłowy wybranych klas.

Klasa *Planner* (Aesculap P).

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Aesculap2P
{
    class Planner
    {
        public string[,] crewData = new string[100, 4];

        public string[,] timetable = new string[32, 10];
        private int[] workHours = new int[100];

        private int crewIndex = 0;
        private int conditionValue1 = 0;

        private string newline = "\n";

        public int days = 0;

        private ProgressBar progressBar1;
        private NumericUpDown numericUpDown1;
        private NumericUpDown numericUpDown2;
        private RichTextBox richTextBox1;
        private SqlCommand sqlCommand1;
        private SqlConnection sqlConnection1;
        private ComboBox comboBox1;

        public int daysCountResult = 0;
        public int employeesCountResult = 0;
        public int vacCheckResult = 0;

        public Planner(ref ProgressBar p, ref NumericUpDown nud1, ref
NumericUpDown nud2, ref RichTextBox rtb1, ref ComboBox cb1, ref SqlCommand
sc1, ref SqlConnection sco1)
        {
            progressBar1 = p;
            numericUpDown1 = nud1;
            numericUpDown2 = nud2;
            richTextBox1 = rtb1;
            comboBox1 = cb1;
            sqlCommand1 = sc1;
            sqlConnection1 = sco1;
```

```
        }

        public void CountDays()
        {

            string monthId = null;

            if (comboBox1.SelectedIndex.Equals(0))
            {
                monthId = System.DateTime.Now.Month.ToString();
                MessageBox.Show(System.DateTime.Now.Month.ToString());
            }
            else
            {
                monthId = comboBox1.SelectedIndex.ToString();
                MessageBox.Show(monthId);
            }

            sqlConnection1.Open();
            sqlCommand1.CommandText = "SELECT COUNT (*) FROM Dyzury WHERE
MONTH (Data) = " + monthId;
            daysCountResult = (int)sqlCommand1.ExecuteScalar();
            sqlConnection1.Close();
        }

        public void CountEmployees()
        {
            sqlConnection1.Open();
            sqlCommand1.CommandText = "SELECT COUNT (*) FROM Lekarze";
            employeesCountResult = (int)sqlCommand1.ExecuteScalar();
            MessageBox.Show(employeesCountResult.ToString());
            sqlConnection1.Close();
        }

        public void CheckVac()
        {

            string monthId = null;

            if (comboBox1.SelectedIndex.Equals(0))
            {
                monthId = System.DateTime.Now.Month.ToString();
                MessageBox.Show(System.DateTime.Now.Month.ToString());
            }
            else
            {
                monthId = comboBox1.SelectedIndex.ToString();
                MessageBox.Show(monthId);
            }

            sqlConnection1.Open();
            sqlCommand1.CommandText = "SELECT COUNT (*) FROM LekarzeUrlopy
WHERE IdMiesiaca = " + monthId;
            vacCheckResult = (int)sqlCommand1.ExecuteScalar();
            sqlConnection1.Close();
        }

        private void SetDays()
        {
            for (int i = 1; i < days+1; i++)
```

```
            {
                timetable[i, 0] = System.Convert.ToString(i);
            }
        }

        private void SetTimetable()
        {
            for (int f = 1; f < days + 1; f++)
            {
                for (int g = 1; g < (int)numericUpDown1.Value + 1; g++)
                {
                    timetable[f, g] = "";
                }
            }
        }

        private void SetWorkHours()
        {
            for (int i = 0; i < employeesCountResult + 1; i++)
            {
                workHours[i] = 0;
            }
        }

        private void SetDayZero()
        {
            for (int h = 0; h < 4; h++)
            {
                timetable[0,h] = "0";
            }
        }

        public void arrange()
        {

            int currentMonth;


            if (comboBox1.SelectedIndex > 0)
            {
                currentMonth = comboBox1.SelectedIndex;

                if (currentMonth == 1 || currentMonth == 3 || currentMonth
== 5 || currentMonth == 7 || currentMonth == 8 || currentMonth == 10 ||
currentMonth == 12)
                {

                    days = 31;
                }
                else if (currentMonth == 2)
                {
                    days = 28;
                }
                else
                {
                    days = 30;
                }
            }
            else
            {
```

```
                currentMonth = System.DateTime.Now.Month;
                if (currentMonth == 1 || currentMonth == 3 || currentMonth
== 5 || currentMonth == 7 || currentMonth == 8 || currentMonth == 10 ||
currentMonth == 12)
                {

                    days = 31;
                }
                else if (currentMonth == 2)
                {
                    days = 28;
                }
                else
                {
                    days = 30;
                }
            }

            SetDays();
            SetTimetable();
            SetWorkHours();
            SetDayZero();

            System.Random generator = new System.Random();

            conditionValue1 = 0;

            progressBar1.Value = 0;

            for (int w = 1; w < days + 1; w++)
            {
                progressBar1.PerformStep();
                richTextBox1.ScrollToCaret();

                for (int u = 1; u < (int)numericUpDown1.Value + 1; u++)
                {

                    crewIndex = generator.Next(employeesCountResult) + 1;

                    for (int z = 1; z < (int)numericUpDown1.Value + 1; z++)
                    {
                        if
((System.Convert.ToString(crewIndex)).Equals(timetable[w, z]) ||
(System.Convert.ToString(crewIndex)).Equals(timetable[w - 1, z]) || (w >
System.Int32.Parse(crewData[crewIndex, 1]) && w <
System.Int32.Parse(crewData[crewIndex, 2])) ||
System.Int32.Parse(crewData[crewIndex, 1]).Equals(w) ||
System.Int32.Parse(crewData[crewIndex, 2]).Equals(w) ||
workHours[crewIndex] >= System.Int32.Parse(crewData[u, 3]))
                        {
                            conditionValue1++;
                        }
                    }

                    if (conditionValue1 == 0)
                    {
                        richTextBox1.AppendText("Dzień "+ w +" Dodaję
lekarza z ID: " + crewIndex + newline);
                        richTextBox1.Refresh();
```

```
                        timetable[w, u] =
System.Convert.ToString(crewIndex);
                        workHours[crewIndex]++;
                    }
                    else
                    {
                        do
                        {
                            conditionValue1 = 0;

                            crewIndex =
generator.Next(employeesCountResult) + 1;
                            for (int z = 1; z < (int)numericUpDown1.Value +
1; z++)
                            {
                                if
((System.Convert.ToString(crewIndex)).Equals(timetable[w, z]) ||
(System.Convert.ToString(crewIndex)).Equals(timetable[w - 1, z]) || (w >
System.Int32.Parse(crewData[crewIndex, 1]) && w <
System.Int32.Parse(crewData[crewIndex, 2])) ||
System.Int32.Parse(crewData[crewIndex, 1]).Equals(w) ||
System.Int32.Parse(crewData[crewIndex, 2]).Equals(w) ||
workHours[crewIndex] >= System.Int32.Parse(crewData[u, 3]))
                                {
                                    conditionValue1++;
                                }
                            }
                        }
                        while (conditionValue1 != 0);

                        richTextBox1.AppendText("Dzień " + w + " Dodaję
lekarza z ID: " + crewIndex + newline);
                        richTextBox1.Refresh();
                        timetable[w, u] =
System.Convert.ToString(crewIndex);
                        workHours[crewIndex]++;
                    }
                }
            }


            System.Console.Out.WriteLine();

            int conditionValue2 = 0;
            int conditionValue3 = 0;

            int m = 0;
            int p = 0;
            int t = 0;

            for (int k = 1; k < employeesCountResult + 1; k++)
            {
                if (workHours[k] < System.Int32.Parse(crewData[k, 3]))
                {
                    conditionValue2++;
                }
            }

            Console.WriteLine("---");
```

```
            while (conditionValue2 != 0)
            {
                conditionValue2 = 0;

                for (t = 1; t < employeesCountResult + 1; t++)
                {

                    if (workHours[t] < System.Int32.Parse(crewData[t, 3]))
                    {

                        if (days == 31)
                        {
                            m = generator.Next(30) + 1;
                        }
                        else if (days == 30)
                        {
                            m = generator.Next(29) + 1;
                        }
                        else
                        {
                            m = generator.Next(27) + 1;
                        }
                        p =
generator.Next(int.Parse(numericUpDown2.Value.ToString())) + 7;

                        while (!(timetable[m, p] == null))
                        {
                            if (days == 31)
                            {
                                m = generator.Next(30) + 1;
                            }
                            else if (days == 30)
                            {
                                m = generator.Next(29) + 1;
                            }
                            else
                            {
                                m = generator.Next(27) + 1;
                            }
                            p =
generator.Next(int.Parse(numericUpDown2.Value.ToString())) + 7;
                        }

                        conditionValue3 = 0;

                        for (int x = 1; x < 6; x++)
                        {
                            if (workHours[t] <
System.Int32.Parse(crewData[t, 3]) &&
(System.Convert.ToString(t).Equals(timetable[m, x])) ||
(System.Convert.ToString(t).Equals(timetable[m - 1, x])) ||
(System.Convert.ToString(t).Equals(timetable[m + 1, x])) || (m >
System.Int32.Parse(crewData[t, 1]) && m < System.Int32.Parse(crewData[t,
2])) || System.Int32.Parse(crewData[t, 1]).Equals(m) ||
System.Int32.Parse(crewData[t, 2]).Equals(m))
                            {
                                conditionValue3++;
                            }
                        }
```

```
                        if (conditionValue3 == 0)
                        {
                            timetable[m, p] = System.Convert.ToString(t);
                            workHours[t]++;
                        }
                    }
                }

                for (int k = 1; k < employeesCountResult + 1; k++)
                {
                    if (workHours[k] < System.Int32.Parse(crewData[k, 3]))
                    {
                        conditionValue2++;
                    }
                }
            }

            if (progressBar1.Value > 0)
            {
                progressBar1.Value = 0;
            }

        }
    }
}
```

Klasa *SplashScreen* (Aesculap R, Aesculap P).

```
using System;
using System.Collections.Generic;
using System.Text;

using System.Drawing;
using System.Drawing.Drawing2D;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Threading;
using System.Diagnostics;
using Microsoft.Win32;

namespace Aesculap2R
{
    public class SplashScreen : System.Windows.Forms.Form
    {
        static SplashScreen ms_frmSplash = null;
        static Thread ms_oThread = null;

        private double m_dblOpacityIncrement = .05;
        private double m_dblOpacityDecrement = .08;
        private const int TIMER_INTERVAL = 50;

        private string m_sStatus;
        private double m_dblCompletionFraction = 0;
        private Rectangle m_rProgress;

        private double m_dblLastCompletionFraction = 0.0;
        private double m_dblPBIncrementPerTimerInterval = .015;

        private bool m_bFirstLaunch = false;
        private DateTime m_dtStart;
        private bool m_bDTSet = false;
        private int m_iIndex = 1;
        private int m_iActualTicks = 0;
        private ArrayList m_alPreviousCompletionFraction;
        private ArrayList m_alActualTimes = new ArrayList();
        private const string REG_KEY_INITIALIZATION = "Initialization";
        private const string REGVALUE_PB_MILISECOND_INCREMENT =
"Increment";
        private const string REGVALUE_PB_PERCENTS = "Percents";

        private System.Windows.Forms.Label lblStatus;
        private System.Windows.Forms.Label lblTimeRemaining;
        private System.Windows.Forms.Timer timer1;
        private System.Windows.Forms.Panel pnlStatus;
        private System.ComponentModel.IContainer components;

        public SplashScreen()
        {
            InitializeComponent();
            this.Opacity = .00;
            timer1.Interval = TIMER_INTERVAL;
            timer1.Start();
            this.ClientSize = this.BackgroundImage.Size;
```

```
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                if (components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(SplashScreen));
            this.lblStatus = new System.Windows.Forms.Label();
            this.pnlStatus = new System.Windows.Forms.Panel();
            this.lblTimeRemaining = new System.Windows.Forms.Label();
            this.timer1 = new System.Windows.Forms.Timer(this.components);
            this.SuspendLayout();
            this.lblStatus.BackColor = System.Drawing.Color.Transparent;
            this.lblStatus.Location = new System.Drawing.Point(79, 160);
            this.lblStatus.Name = "lblStatus";
            this.lblStatus.Size = new System.Drawing.Size(237, 14);
            this.lblStatus.TabIndex = 0;
            this.pnlStatus.BackColor = System.Drawing.Color.Transparent;
            this.pnlStatus.Location = new System.Drawing.Point(82, 141);
            this.pnlStatus.Name = "pnlStatus";
            this.pnlStatus.Size = new System.Drawing.Size(237, 10);
            this.pnlStatus.TabIndex = 1;
            this.pnlStatus.Paint += new
System.Windows.Forms.PaintEventHandler(this.pnlStatus_Paint);
            this.lblTimeRemaining.BackColor =
System.Drawing.Color.Transparent;
            this.lblTimeRemaining.Location = new System.Drawing.Point(79,
178);
            this.lblTimeRemaining.Name = "lblTimeRemaining";
            this.lblTimeRemaining.Size = new System.Drawing.Size(237, 16);
            this.lblTimeRemaining.TabIndex = 2;
            this.lblTimeRemaining.Text = "Pozostało sekund: ";
            this.timer1.Tick += new System.EventHandler(this.timer1_Tick);
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.BackColor = System.Drawing.Color.LightGray;
            this.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("$this.BackgroundImage")));
            this.ClientSize = new System.Drawing.Size(400, 200);
            this.Controls.Add(this.lblTimeRemaining);
            this.Controls.Add(this.pnlStatus);
            this.Controls.Add(this.lblStatus);
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.None;
            this.Name = "SplashScreen";
            this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "SplashScreen";
```


```
            this.DoubleClick += new
System.EventHandler(this.SplashScreen_DoubleClick);
            this.ResumeLayout(false);

        }
        #endregion

        static public void ShowSplashScreen()
        {
            if (ms_frmSplash != null)
                return;
            ms_oThread = new Thread(new
ThreadStart(SplashScreen.ShowForm));
            ms_oThread.IsBackground = true;
            ms_oThread.ApartmentState = ApartmentState.STA;
            ms_oThread.Start();
        }

        static public SplashScreen SplashForm
        {
            get
            {
                return ms_frmSplash;
            }
        }

        static private void ShowForm()
        {
            ms_frmSplash = new SplashScreen();
            Application.Run(ms_frmSplash);
        }

        static public void CloseForm()
        {
            if (ms_frmSplash != null && ms_frmSplash.IsDisposed == false)
            {

                ms_frmSplash.m_dblOpacityIncrement = -
                   ms_frmSplash.m_dblOpacityDecrement;
            }
            ms_oThread = null;
            ms_frmSplash = null;
        }

        static public void SetStatus(string newStatus)
        {
            SetStatus(newStatus, true);
        }

        static public void SetStatus(string newStatus, bool setReference)
        {
            if (ms_frmSplash == null)
                return;
            ms_frmSplash.m_sStatus = newStatus;
            if (setReference)
                ms_frmSplash.SetReferenceInternal();
        }

        static public void SetReferencePoint()
        {
```

```
            if (ms_frmSplash == null)
                return;
            ms_frmSplash.SetReferenceInternal();

        }

        private void SetReferenceInternal()
        {
            if (m_bDTSet == false)
            {
                m_bDTSet = true;
                m_dtStart = DateTime.Now;
                ReadIncrements();
            }
            double dblMilliseconds = ElapsedMilliSeconds();
            m_alActualTimes.Add(dblMilliseconds);
            m_dblLastCompletionFraction = m_dblCompletionFraction;
            if (m_alPreviousCompletionFraction != null
                && m_iIndex < m_alPreviousCompletionFraction.Count)
                m_dblCompletionFraction =
                    (double)m_alPreviousCompletionFraction[m_iIndex++];
            else
                m_dblCompletionFraction = (m_iIndex > 0) ? 1 : 0;
        }

        private double ElapsedMilliSeconds()
        {
            TimeSpan ts = DateTime.Now - m_dtStart;
            return ts.TotalMilliseconds;
        }

        private void ReadIncrements()
        {
            string sPBIncrementPerTimerInterval =
                   RegistryAccess.GetStringRegistryValue(
                       REGVALUE_PB_MILISECOND_INCREMENT, "0.0015");
            double dblResult;

            if (Double.TryParse(sPBIncrementPerTimerInterval,
                    System.Globalization.NumberStyles.Float,
                    System.Globalization.NumberFormatInfo.InvariantInfo,
                    out dblResult))
                m_dblPBIncrementPerTimerInterval = dblResult;
            else
                m_dblPBIncrementPerTimerInterval = .0015;

            string sPBPreviousPctComplete =
RegistryAccess.GetStringRegistryValue(
                  REGVALUE_PB_PERCENTS, "");

            if (sPBPreviousPctComplete != "")
            {
                string[] aTimes = sPBPreviousPctComplete.Split(null);
                m_alPreviousCompletionFraction = new ArrayList();

                for (int i = 0; i < aTimes.Length; i++)
                {
                    double dblVal;
                    if (Double.TryParse(aTimes[i],
                            System.Globalization.NumberStyles.Float,
```

```
System.Globalization.NumberFormatInfo.InvariantInfo,
                            out dblVal))
                        m_alPreviousCompletionFraction.Add(dblVal);
                    else
                        m_alPreviousCompletionFraction.Add(1.0);
                }
            }
            else
            {
                m_bFirstLaunch = true;
                lblTimeRemaining.Text = "";
            }
        }

        private void StoreIncrements()
        {
            string sPercent = "";
            double dblElapsedMilliseconds = ElapsedMilliSeconds();
            for (int i = 0; i < m_alActualTimes.Count; i++)
                sPercent += ((double)m_alActualTimes[i] /
                      dblElapsedMilliseconds).ToString("0.####",
                      System.Globalization.NumberFormatInfo.InvariantInfo)
+ " ";

            RegistryAccess.SetStringRegistryValue(
                REGVALUE_PB_PERCENTS, sPercent);

            m_dblPBIncrementPerTimerInterval = 1.0 /
(double)m_iActualTicks;
            RegistryAccess.SetStringRegistryValue(
                 REGVALUE_PB_MILISECOND_INCREMENT,
                 m_dblPBIncrementPerTimerInterval.ToString("#.000000",
                 System.Globalization.NumberFormatInfo.InvariantInfo));
        }

        private void timer1_Tick(object sender, System.EventArgs e)
        {
            lblStatus.Text = m_sStatus;

            if (m_dblOpacityIncrement > 0)
            {
                m_iActualTicks++;
                if (this.Opacity < 1)
                    this.Opacity += m_dblOpacityIncrement;
            }
            else
            {
                if (this.Opacity > 0)
                    this.Opacity += m_dblOpacityIncrement;
                else
                {
                    StoreIncrements();
                    this.Close();
                }
            }
            if (m_bFirstLaunch == false && m_dblLastCompletionFraction
                < m_dblCompletionFraction)
            {
```

```
                m_dblLastCompletionFraction +=
m_dblPBIncrementPerTimerInterval;
                int width = (int)Math.Floor(
                   pnlStatus.ClientRectangle.Width *
m_dblLastCompletionFraction);
                int height = pnlStatus.ClientRectangle.Height;
                int x = pnlStatus.ClientRectangle.X;
                int y = pnlStatus.ClientRectangle.Y;
                if (width > 0 && height > 0)
                {
                    m_rProgress = new Rectangle(x, y, width, height);
                    pnlStatus.Invalidate(m_rProgress);
                    int iSecondsLeft = 1 + (int)(TIMER_INTERVAL *
                      ((1.0 - m_dblLastCompletionFraction) /
                        m_dblPBIncrementPerTimerInterval)) / 1000;
                    if (iSecondsLeft == 1)
                        lblTimeRemaining.Text = string.Format("Gotowe!");
                    else
                        lblTimeRemaining.Text = string.Format("Pozostało
sekund: {0}",
                          iSecondsLeft);

                }
            }
        }

        private void pnlStatus_Paint(object sender,
             System.Windows.Forms.PaintEventArgs e)
        {
            if (m_bFirstLaunch == false && e.ClipRectangle.Width > 0
                  && m_iActualTicks > 1)
            {
                LinearGradientBrush brBackground =
                  new LinearGradientBrush(m_rProgress,
                                          Color.FromArgb(100, 100, 100),
                                          Color.FromArgb(150, 150, 255),
                                          LinearGradientMode.Horizontal);
                e.Graphics.FillRectangle(brBackground, m_rProgress);
            }
        }

        private void SplashScreen_DoubleClick(object sender,
System.EventArgs e)
        {
            CloseForm();
        }
    }

    public class RegistryAccess
    {
        private const string SOFTWARE_KEY = "Software";
        private const string COMPANY_NAME = "MyCompany";
        private const string APPLICATION_NAME = "MyApplication";

        static public string GetStringRegistryValue(string key,
            string defaultValue)
        {
            RegistryKey rkCompany;
            RegistryKey rkApplication;
```

```
            rkCompany = Registry.CurrentUser.OpenSubKey(SOFTWARE_KEY,
                 false).OpenSubKey(COMPANY_NAME, false);
            if (rkCompany != null)
            {
                rkApplication = rkCompany.OpenSubKey(APPLICATION_NAME,
true);
                if (rkApplication != null)
                {
                    foreach (string sKey in rkApplication.GetValueNames())
                    {
                        if (sKey == key)
                        {
                            return (string)rkApplication.GetValue(sKey);
                        }
                    }
                }
            }
            return defaultValue;
        }

        static public void SetStringRegistryValue(string key,
             string stringValue)
        {
            RegistryKey rkSoftware;
            RegistryKey rkCompany;
            RegistryKey rkApplication;

            rkSoftware = Registry.CurrentUser.OpenSubKey(SOFTWARE_KEY,
true);
            rkCompany = rkSoftware.CreateSubKey(COMPANY_NAME);
            if (rkCompany != null)
            {
                rkApplication = rkCompany.CreateSubKey(APPLICATION_NAME);
                if (rkApplication != null)
                {
                    rkApplication.SetValue(key, stringValue);
                }
            }
        }

    }
}
```

4. Instrukcja obsługi.

4.1. Skrócony opis działania poszczególnych modułów systemu Aesculap.

Użytkownicy systemu Aesculap podzieleni są na trzy grupy posiadające oddzielne identyfikatory. Grupa użytkowników z identyfikatorem 1 ma uprawnienia do używania modułu Aesculap R, grupa użytkowników z identyfikatorem 2 ma z kolei uprawnienia do używania modułu Aesculap P natomiast grupa użytkowników z identyfikatorem 3 ma uprawnienia do używania modułu Aesculap H.

Moduł Aesculap R jest przeznaczony do obsługi działu rejestracji szpitala. Przy jego użyciu użytkownik może rejestrować pacjentów i edytować oraz przeglądać dane zarejestrowanych już pacjentów w tabelach wyświetlających dane ogólne i szczegółowe. Jeżeli szczegółowe dane pacjenta zostały zdefiniowane, to nie ma możliwości ponownego ich definiowania. Można edytować dane zarejestrowanych pacjentów. Rejestracja przyjęć pacjentów odbywa się na zasadzie wyboru odpowiedniego pacjenta z tabeli zarejestrowanych pacjentów i wyboru daty przyjęcia. Podobnie jest z wypisami. Tutaj użytkownik wybiera pacjenta i ustala datę wypisu. Ustalanie wizyt lekarskich polega na wyborze pacjenta i lekarza z tabel z danymi pacjentów i lekarzy oraz ustaleniu daty wizyty. Ustalanie zabiegów jest analogiczne do ustalania wizyt lekarskich i obejmuje wybór pacjenta i sali z tabel z danymi pacjentów i sal i ustalenie daty zabiegu. Kompletowanie składu lekarzy na zabieg polega na wyborze odpowiedniego zabiegu z tabeli z danymi zabiegów i maksymalnie trzech lekarzy, posługując się tabelą z danymi lekarzy.

Moduł Aesculap P jest przeznaczony do planowania grafiku dyżurów lekarskich. Przed rozpoczęciem planowania użytkownik definiuje liczbę lekarzy potrzebnych do sprawowania dyżuru w ciągu jednego dnia, liczbę lekarzy dodatkowych oraz miesiąc, na który harmonogram ma zostać zaplanowany. Pasek postępu w dolnej części okna obrazuje postęp procesu planowania grafiku. Wyświetlane są także na bieżąco

rezultaty planowania dla kolejnych dni wybranego miesiąca. Można je również wydrukować. Planowanie na dany dzień przebiega tak, aby w dniu poprzedzającym ten dzień oraz następnym identyfikatory lekarzy nie powtarzały się. Identyfikatory lekarzy nie mogą się także powtarzać w obrębie zespołu lekarzy pełniących dyżur danego dnia. Ponadto, każdy lekarz ma zdefiniowaną liczbę godzin pracy oraz określoną liczbę dni urlopu w miesiącu. Aesculap P tak planuje grafik, aby godziny pracy lekarzy zostały w pełni wykorzystane. Jeżeli lekarz nie ma wykorzystanych wszystkich godzin pracy, to poza normalnymi dniami pracy może pełnić dyżur jako lekarz dodatkowy. Planowanie odbywa się także ze zwróceniem uwagi na zdefiniowane urlopy, tzn. lekarz nie może zostać przydzielony do pełnienia dyżuru w okresie oznaczonym dla niego jako urlop, Aesculap P pozwala użytkownikowi na przeglądnie zaplanowanych dyżurów w specjalnej tabeli i ich edycję z poziomu tej tabeli. Użytkownik może także edytować liczbę wymaganych godzin pracy dla poszczególnych lekarzy i danych urlopów dla poszczególnych lekarzy.

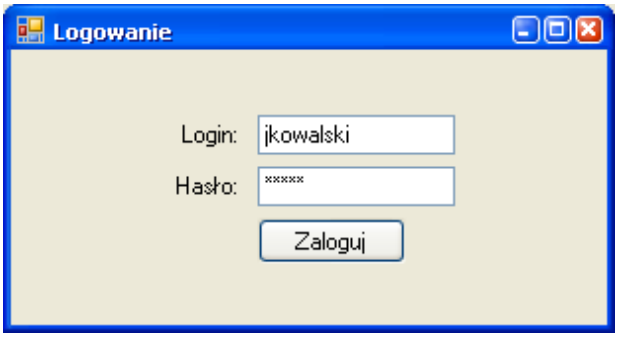
Moduł Aesculap H jest przeznaczony dla lekarzy. Za jego pomocą mogą oni przeglądać ustalone wizyty lekarskie, a także zabiegi oraz przeglądać i ustalać czynności do wykonania podczas zabiegu. Przy użyciu Aesculap H, lekarze mogą także przeglądać przyjęcia pacjentów oraz przeglądać i edytować dane rozpoznań powiązanych z wybranymi przyjęciami pacjentów, W przypadku śmierci pacjenta, lekarz może zapisać informacje o zgonie . Może także przeglądać dane zgonów innych pacjentów. Lekarz może wykorzystać listę leków i chorób ze szczegółowymi danymi na ich temat podczas wystawiania diagnozy. Obserwacje zmian stanu zdrowia pacjenta mogą również być rejestrowane. Podobnie jest z historią rozpoznanych chorób pacjenta i historią użytych leków. Lekarz ma także do dyspozycji tabelę z grafikiem dyżurów, który może przeglądać. Istnieje także możliwość synchronizacji danych ze stacjonarnym serwerem bazy danych w dowolnym momencie.

4.2. Ekran startowy (Aesculap R, P).

Rys. 5. Ekran startowy modułu Aesculap R.

Rys. 6. Ekran startowy modułu Aesculap P.

4.3. Logowanie (Aesculap R, P, H).



Rys. 7. Okno logowania modułów Aesculap R i Aesculap P.

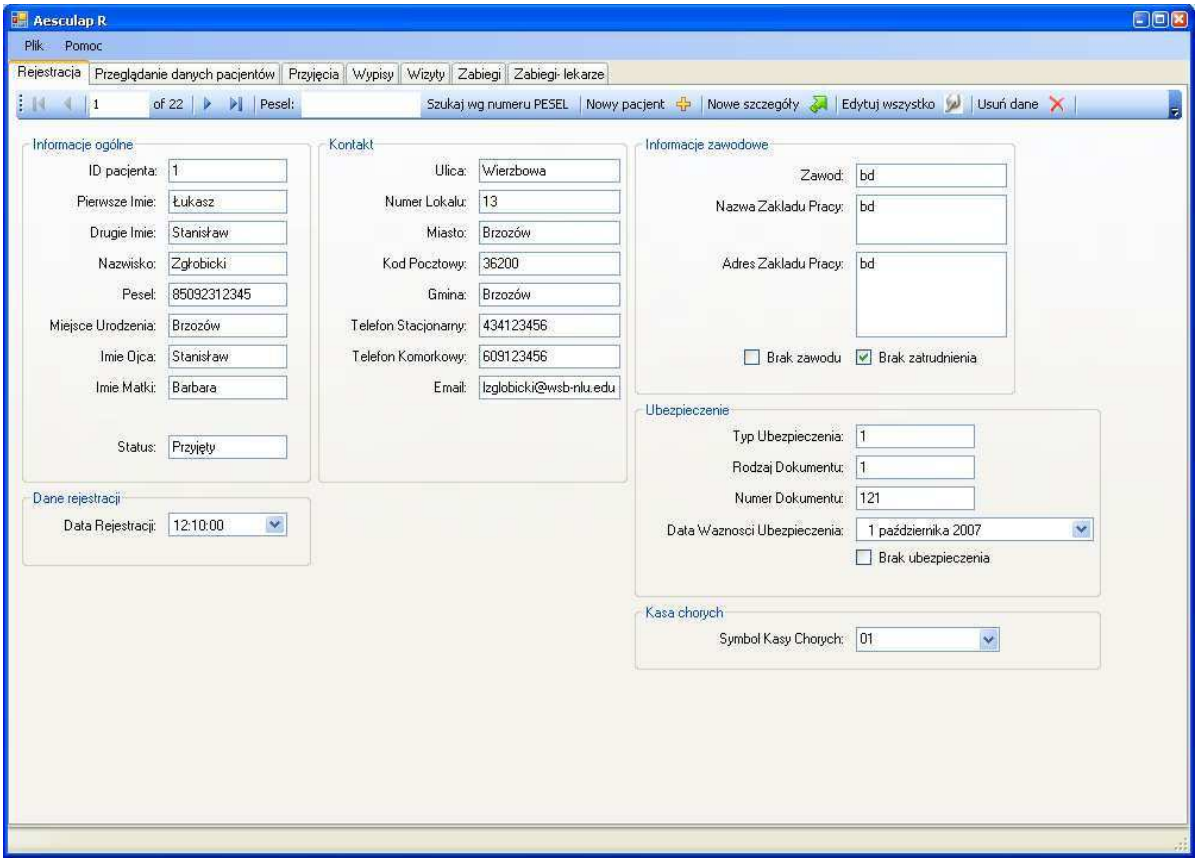
Rys. 8. Zakładka logowania modułu Aesculap H.

4.4. Informacje o programie (Aesculap R, P, H).

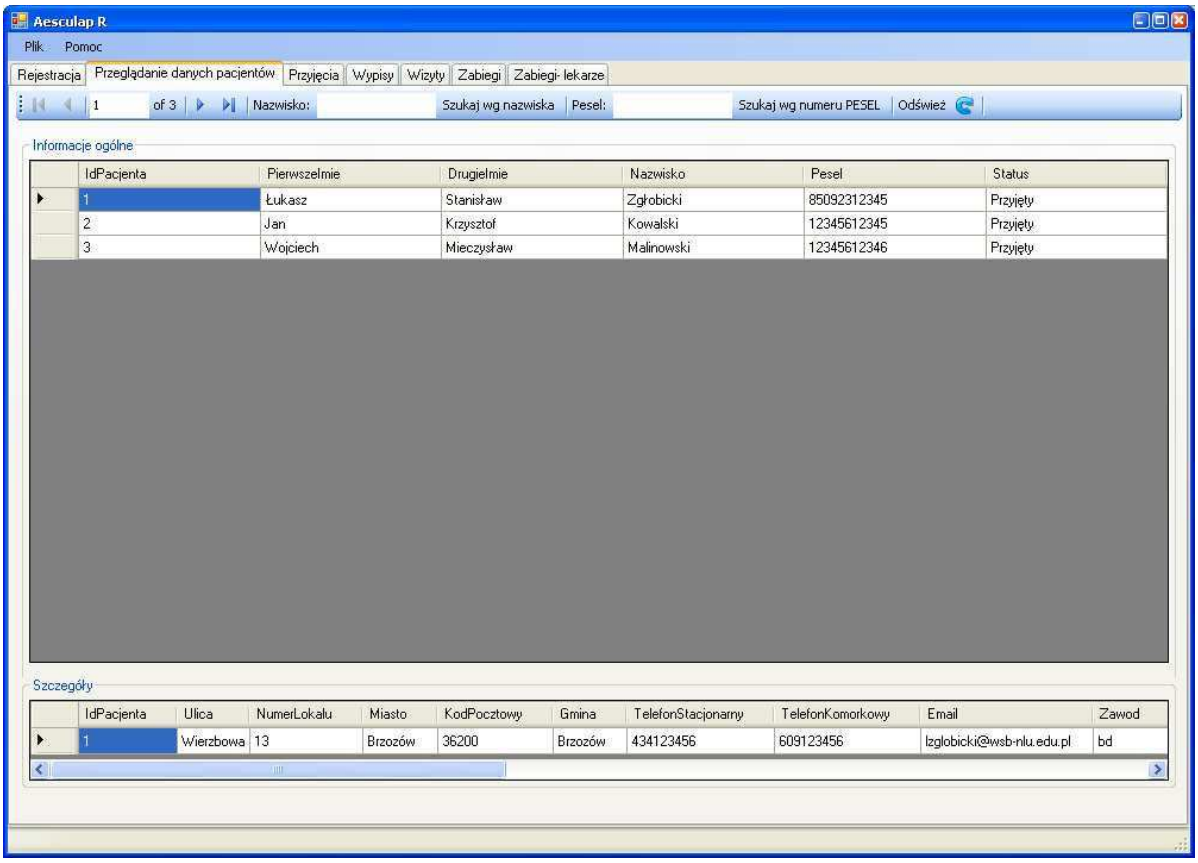
Rys. 9. Okno informacji o programie modułu Aesculap R. Okno informacji o progamie modułu Aesculap P wygląda identycznie.

Rys. 10. Zakładka infomacji o programie modułu Aesculap H.

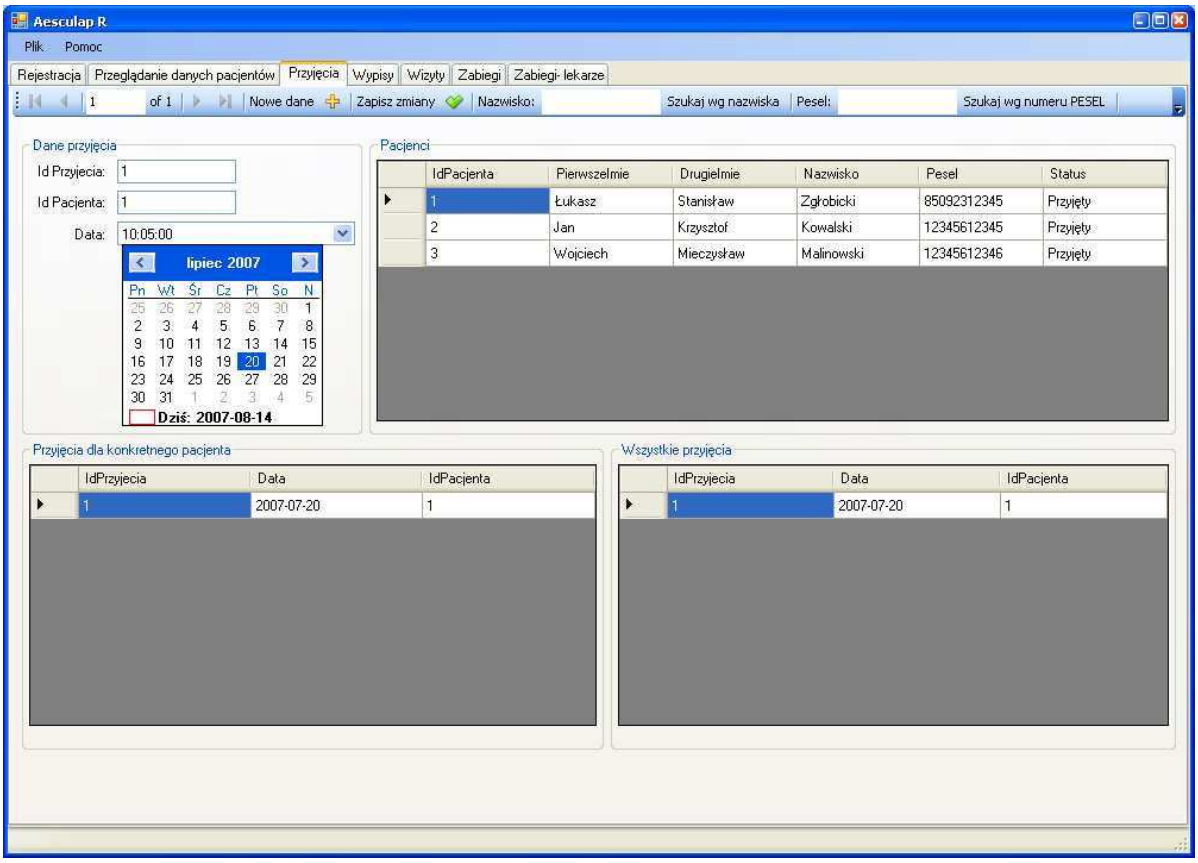
4.5. Pozostałe funkcje.



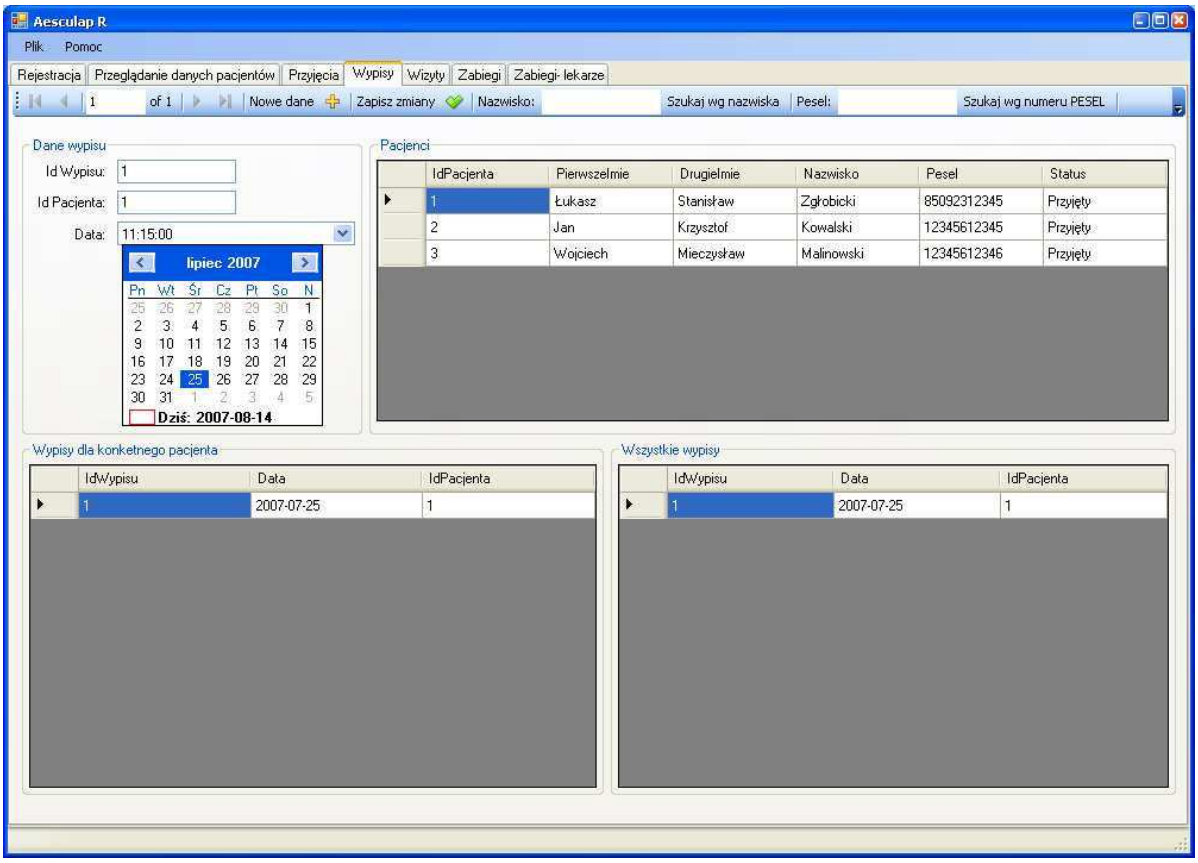
Rys. 11. Zakładka rejestracji pacjentów modułu Aesculap H z możliwością edycji ogólnych i szczegółowych danych pacjenta.



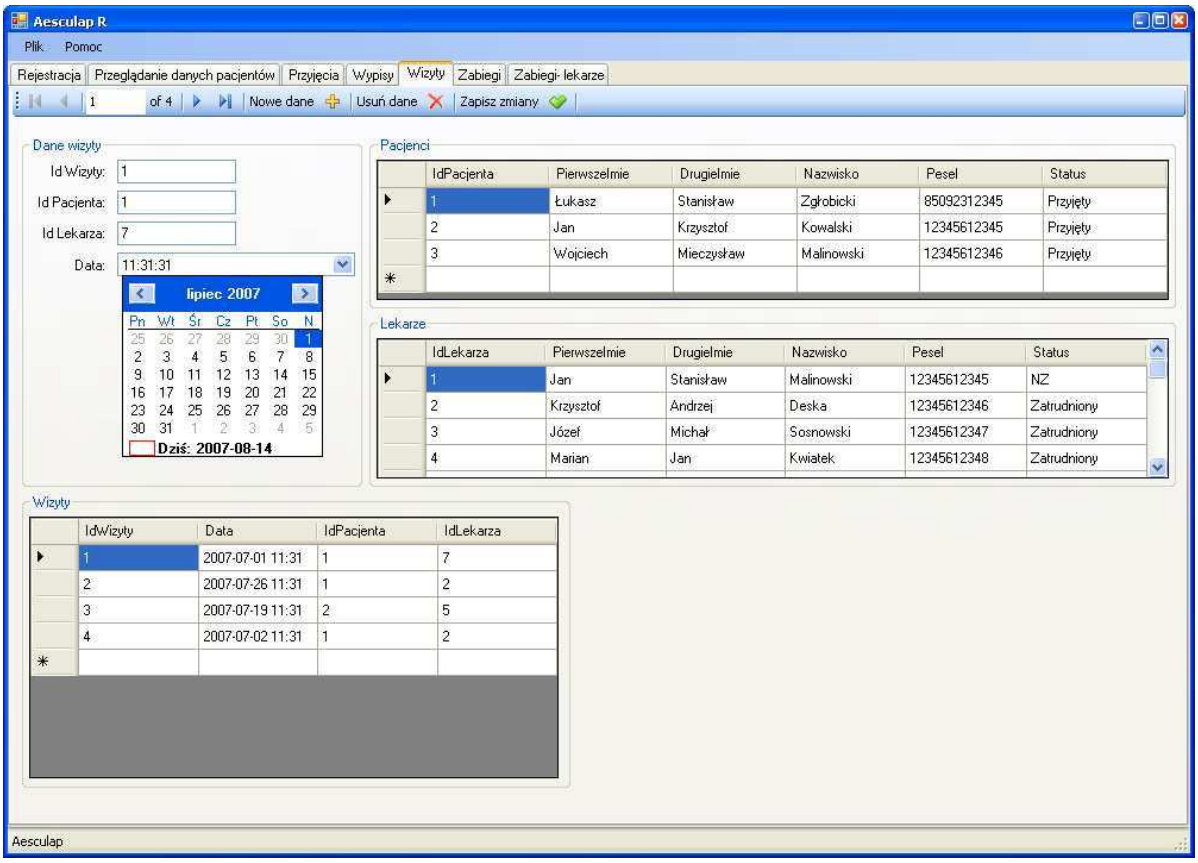
Rys. 12. Zakładka przeglądania danych zarejestrowanych pacjentów modułu Aesculap R – widok danych ogólnych i szczegółowych.



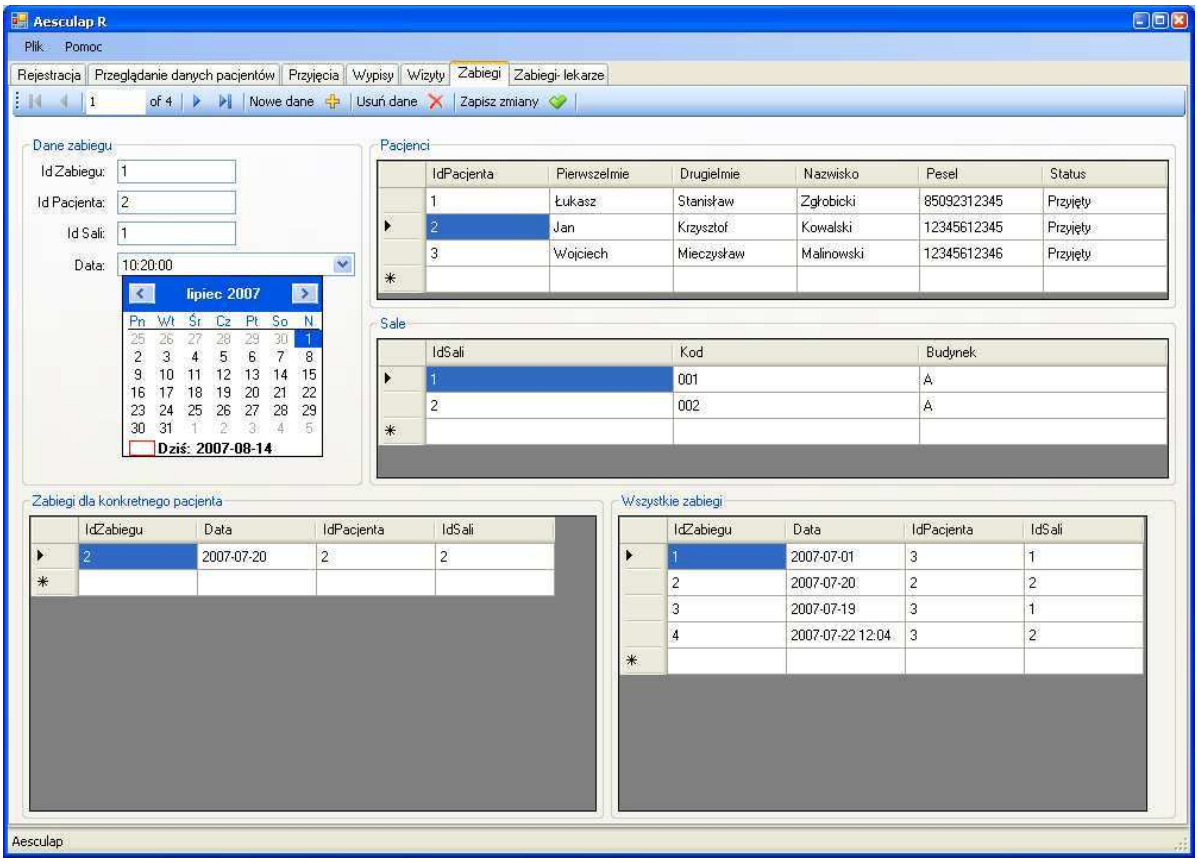
Rys. 13. Zakładka edycji i przeglądania danych przyjęć pacjentów modułu Aesculap R.



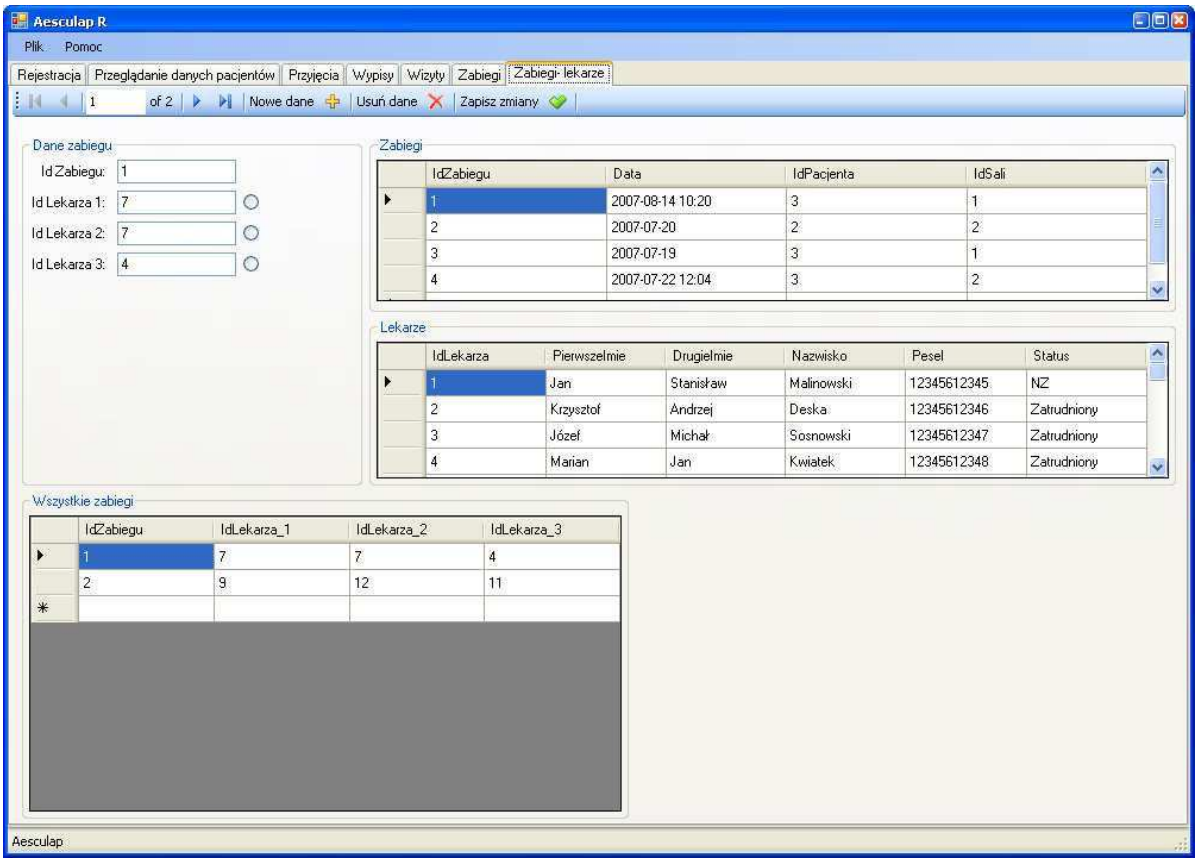
Rys. 14. Zakładka edycji i przeglądania danych wypisów pacjentów modułu Aesculap R.



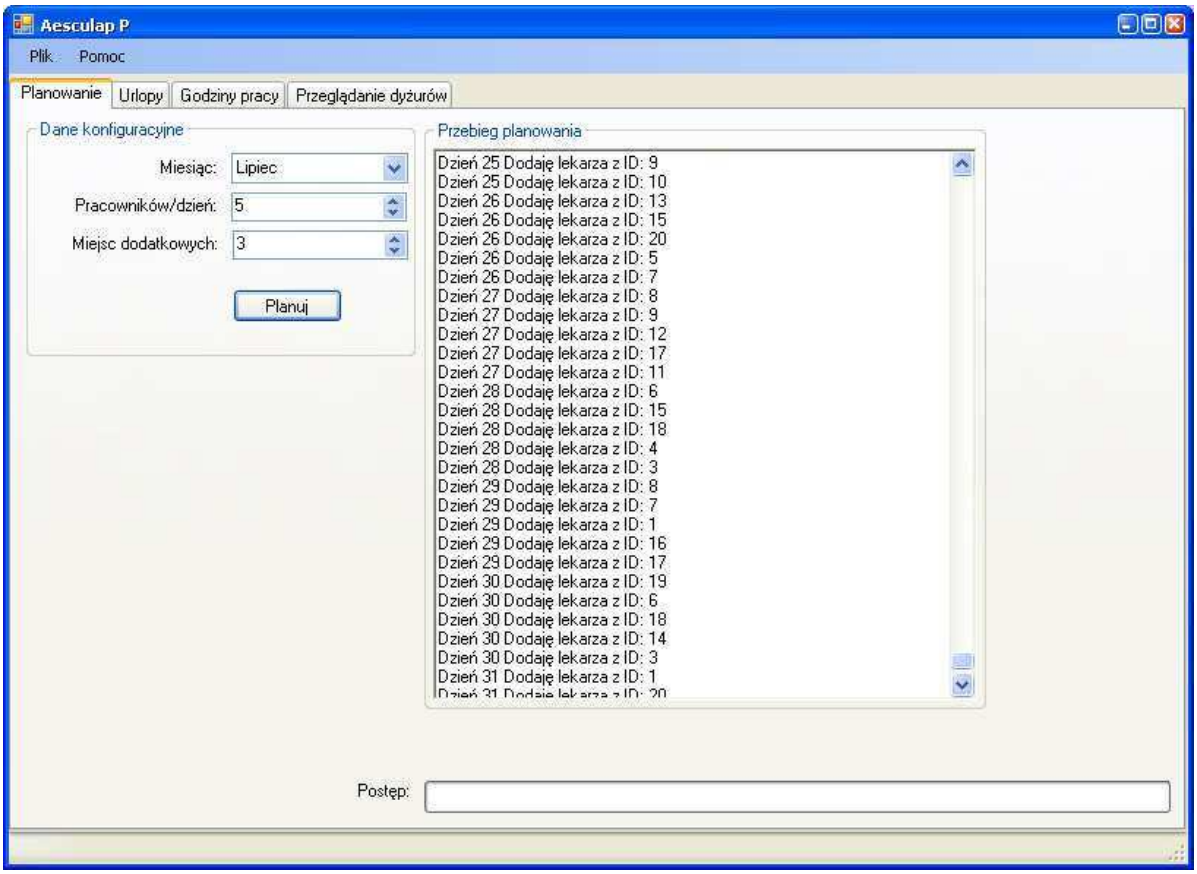
Rys. 15. Zakładka edycji i przeglądania danych wizyt lekarskich modułu Aesculap R.



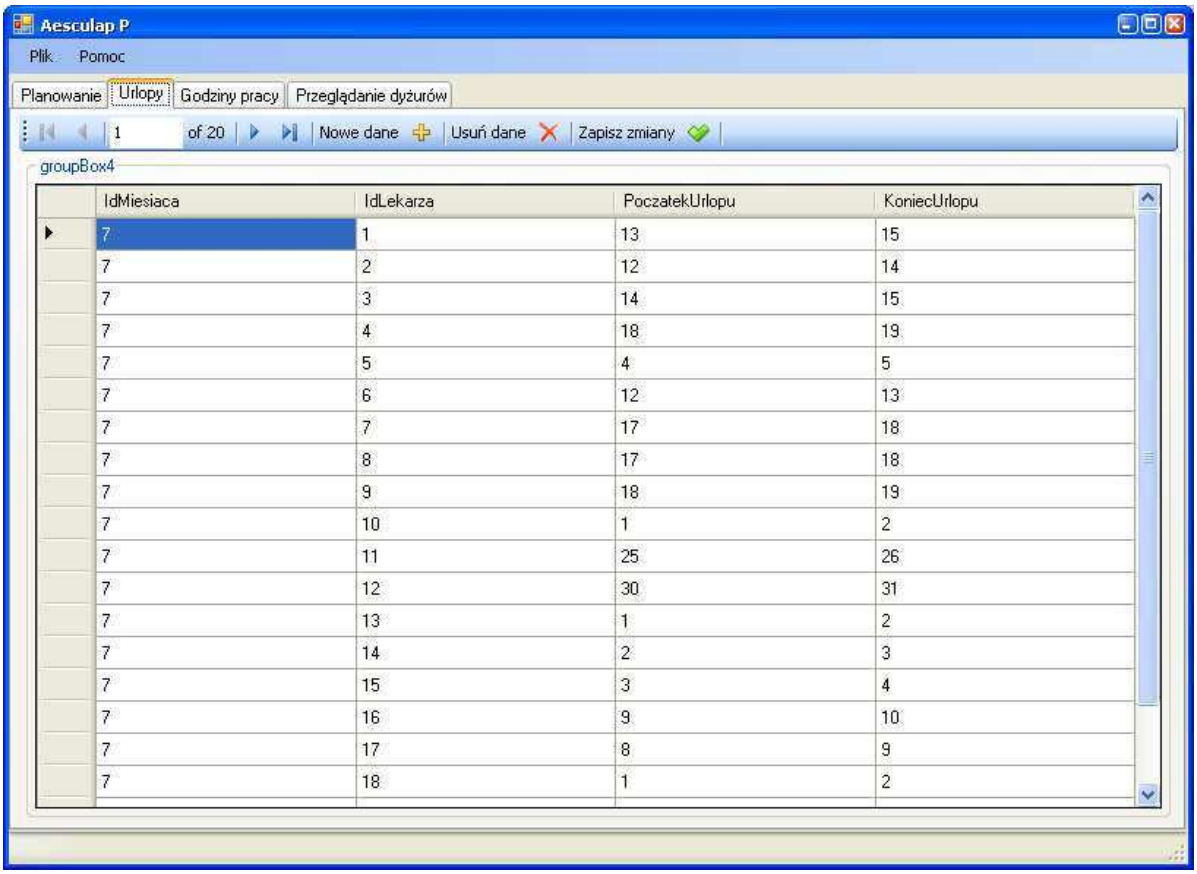
Rys. 16. Zakładka edycji i przeglądania danych zabiegów modułu Aesculap R.



Rys. 17. Zakładka edycji i przeglądania danych szczegółowych zaplanowanych modułu Aesculap R.

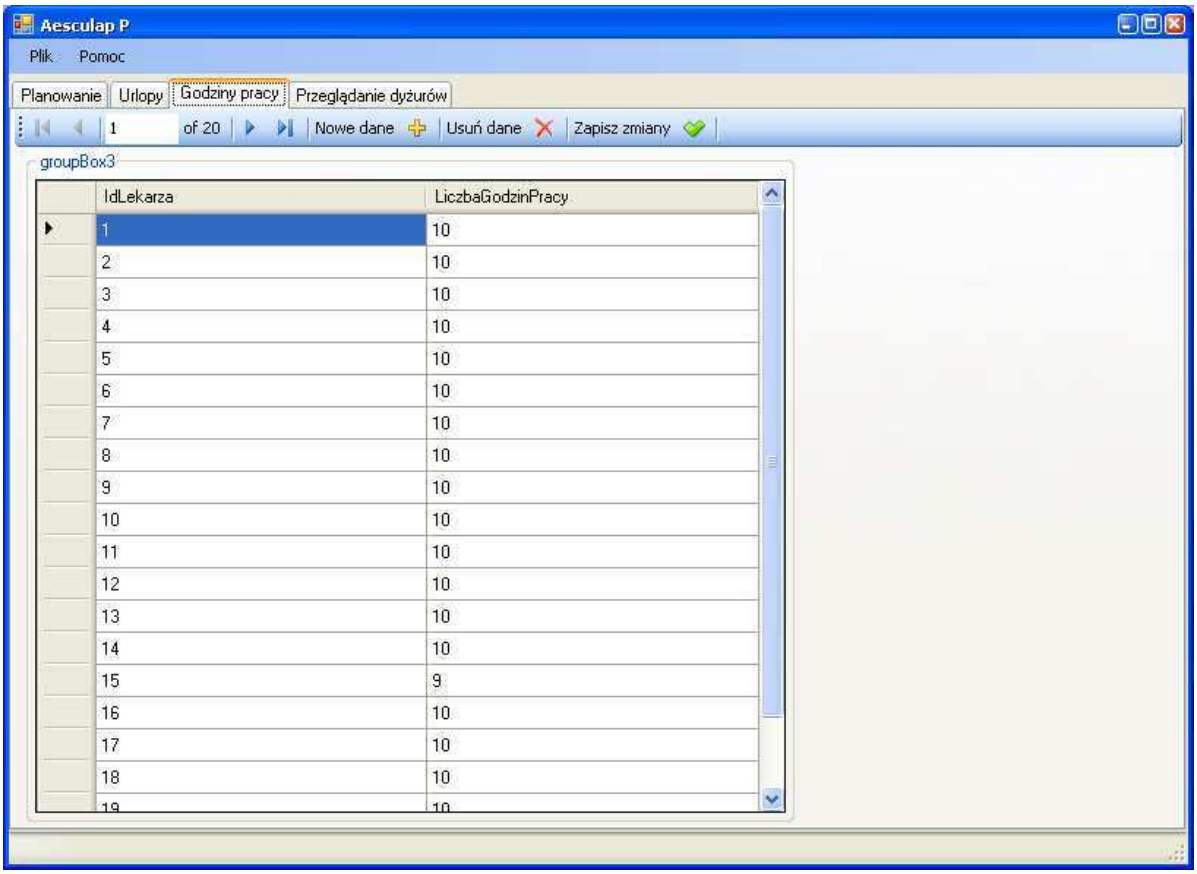


Rys. 18. Zakładka planowania grafiku dyżurów modułu Aesculap P.

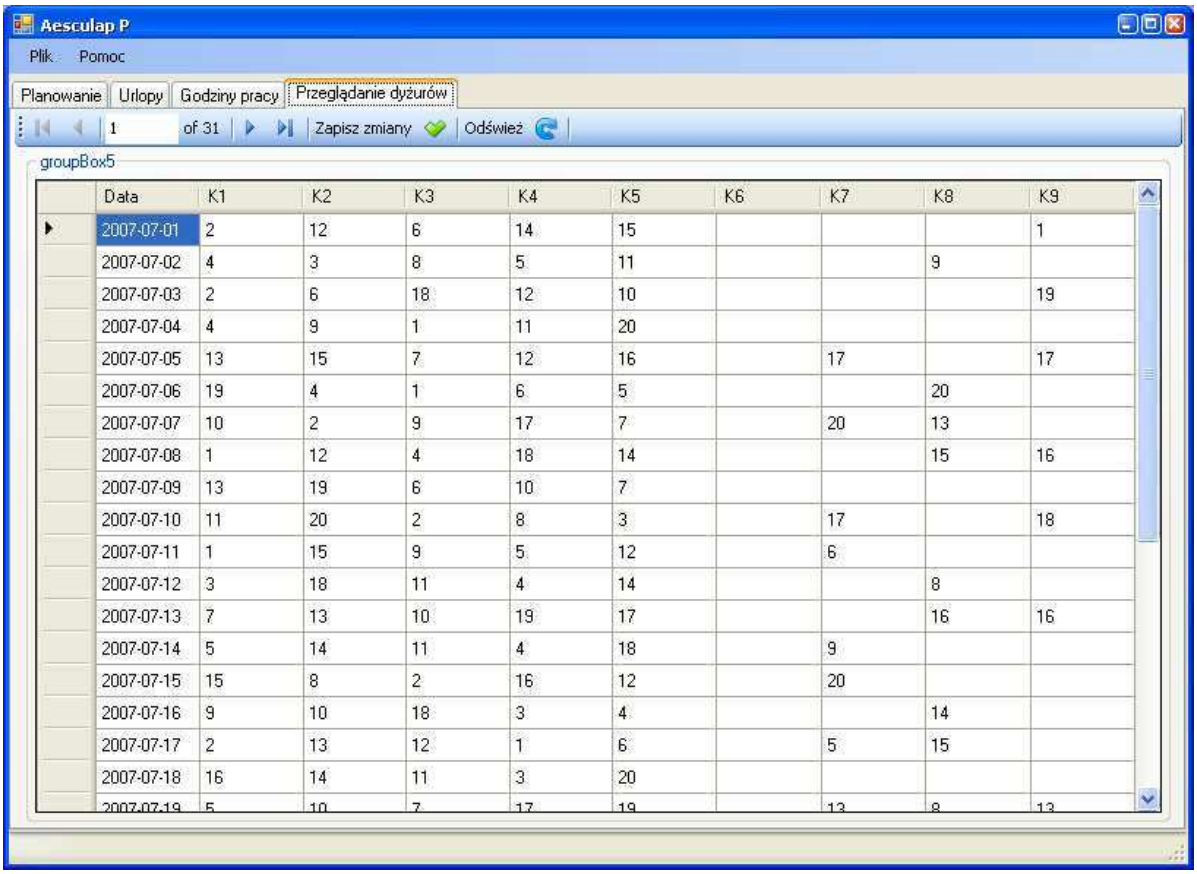


IdMiesiaca	IdLekarza	PoczatekUrlopu	KoniecUrlopu
7	1	13	15
7	2	12	14
7	3	14	15
7	4	18	19
7	5	4	5
7	6	12	13
7	7	17	18
7	8	17	18
7	9	18	19
7	10	1	2
7	11	25	26
7	12	30	31
7	13	1	2
7	14	2	3
7	15	3	4
7	16	9	10
7	17	8	9
7	18	1	2

Rys. 19. Zakładka edycji i przeglądania danych urlopów lekarzy modułu Aesculap P.

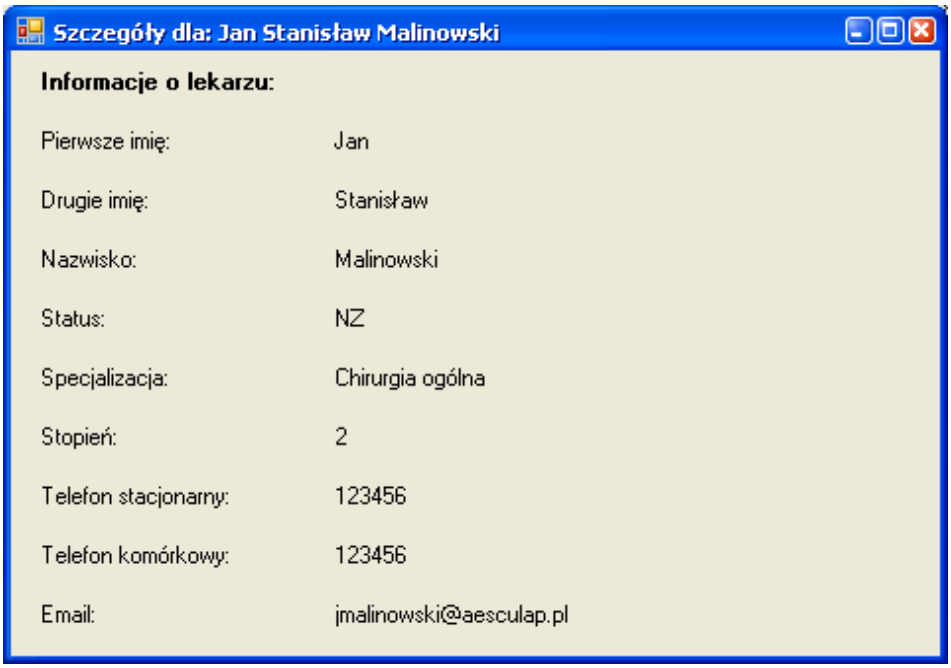


Rys. 20. Zakładka edycji i przeglądania danych o godzinach pracy lekarzy modułu Aesculap P.



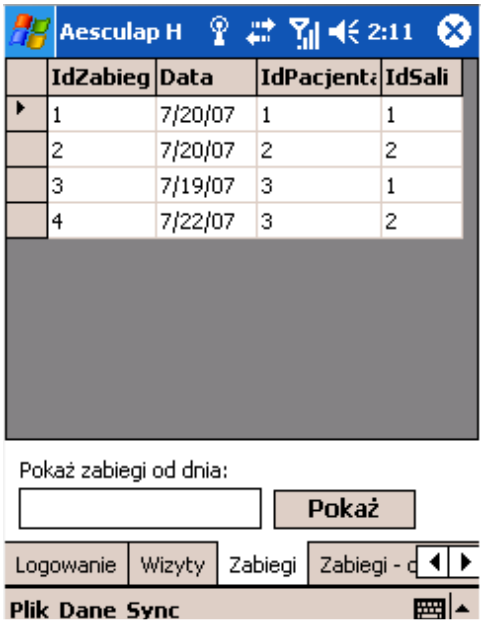
Data	K1	K2	K3	K4	K5	K6	K7	K8	K9
2007-07-01	2	12	6	14	15				1
2007-07-02	4	3	8	5	11			9	
2007-07-03	2	6	18	12	10				19
2007-07-04	4	9	1	11	20				
2007-07-05	13	15	7	12	16		17		17
2007-07-06	19	4	1	6	5			20	
2007-07-07	10	2	9	17	7		20	13	
2007-07-08	1	12	4	18	14			15	16
2007-07-09	13	19	6	10	7				
2007-07-10	11	20	2	8	3		17		18
2007-07-11	1	15	9	5	12		6		
2007-07-12	3	18	11	4	14			8	
2007-07-13	7	13	10	19	17			16	16
2007-07-14	5	14	11	4	18		9		
2007-07-15	15	8	2	16	12		20		
2007-07-16	9	10	18	3	4			14	
2007-07-17	2	13	12	1	6		5	15	
2007-07-18	16	14	11	3	20				
2007-07-19	5	10	7	17	19		13	8	13

Rys. 21. Zakładka edycji i przeglądania grafiku dyżurów lekarskich modułu Aesculap P.

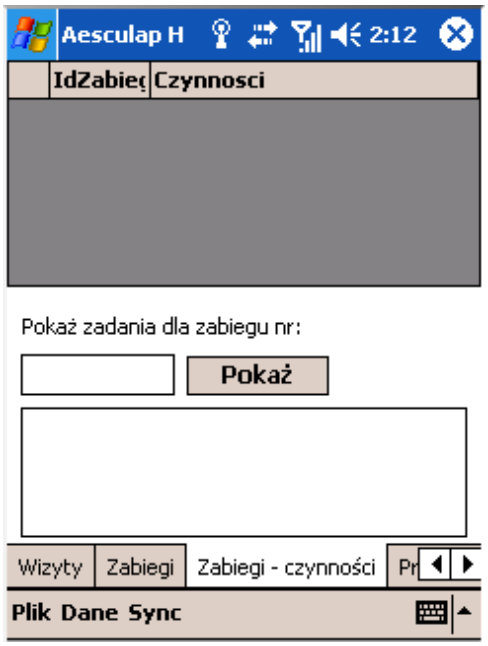


Rys. 22. Okno informacji o lekarzu modułu Aesculap P.

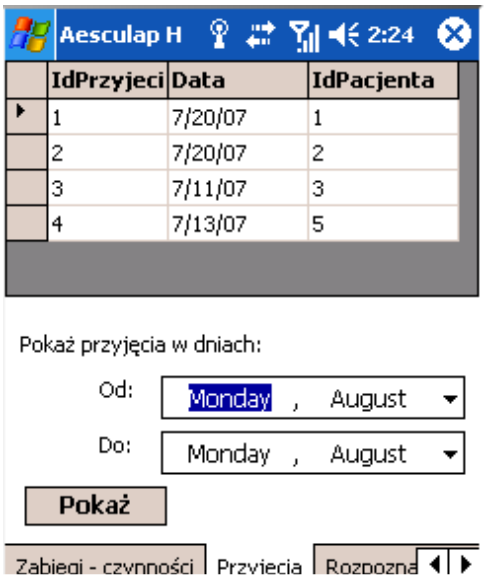
Rys. 23. Zakładka przeglądania danych wizyt lekarskich modułu Aesculap H.



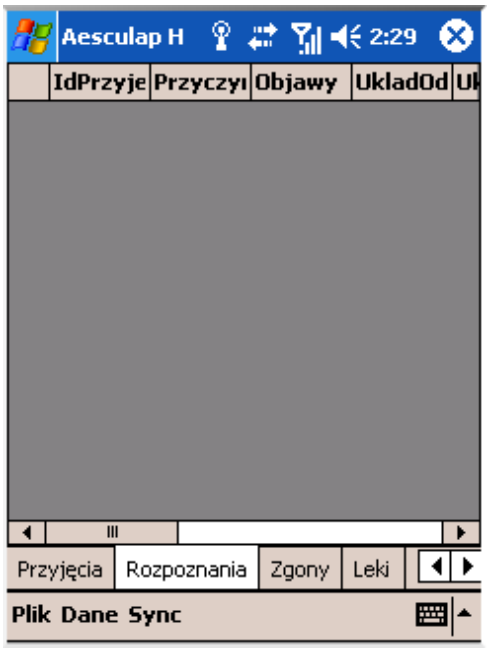
Rys. 24. Zakładka przeglądania danych zabiegów modułu Aesculap H.



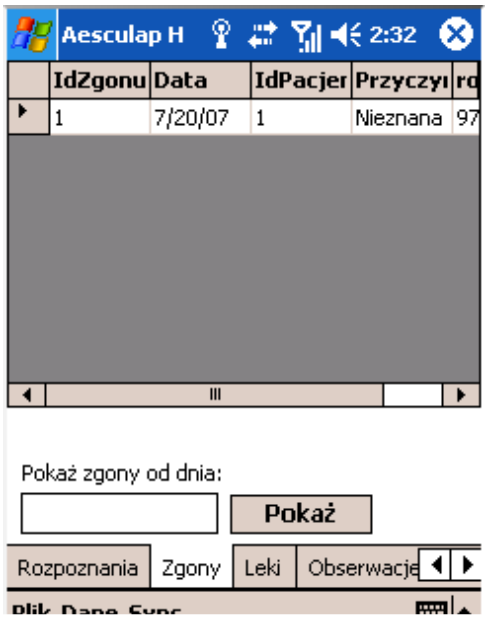
Rys. 25. Zakładka edycji i przeglądania danych o czynnościach do wykonania podczas zabiegu modułu Aesculap H.



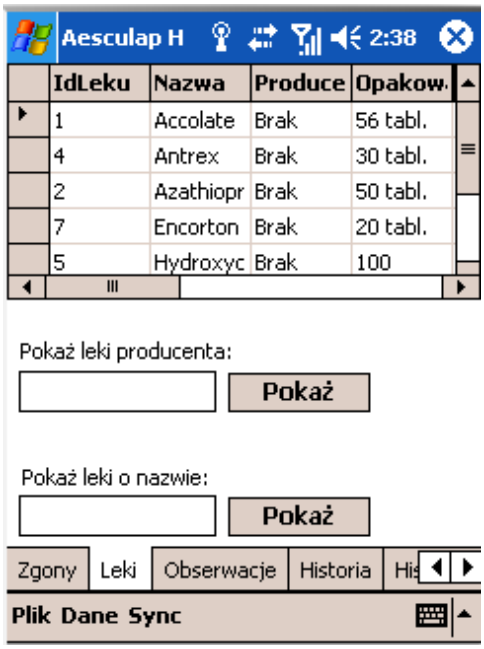
Rys. 26. Zakładka przeglądania danych przyjęć pacjentów do szpitala modułu Aesculap H.



Rys. 27. Zakładka edycji i przeglądania danych rozpoznań modułu Aesculap H.



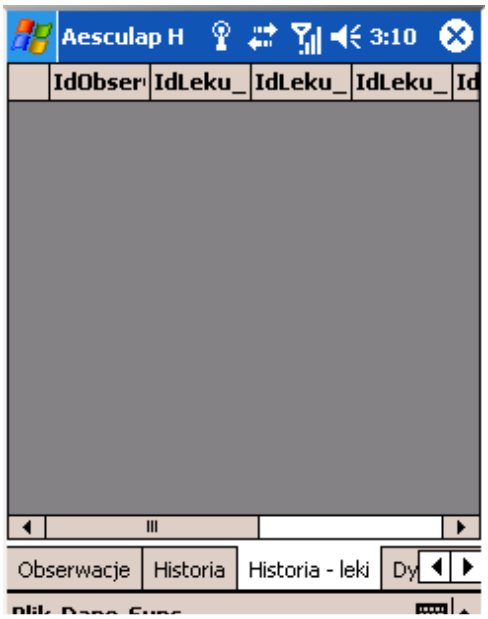
Rys. 28. Zakładka edycji i przeglądania danych o zgonach pacjentów modułu Aesculap H.



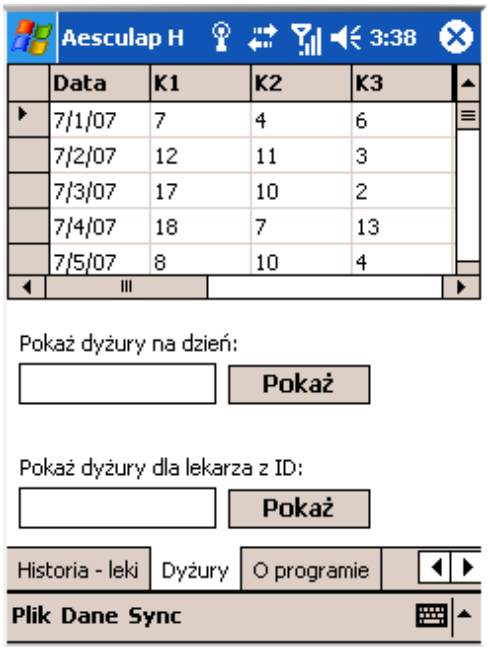
Rys. 29. Zakładka przeglądania danych leków modułu Aesculap H.

Rys. 30. Zakładka edycji i przeglądania danych obserwacji stanu zdrowia pacjentów modułu Aesculap H.

Rys. 31. Zakładka edycji i przeglądania danych historii chorób pacjentów modułu Aesculap H.

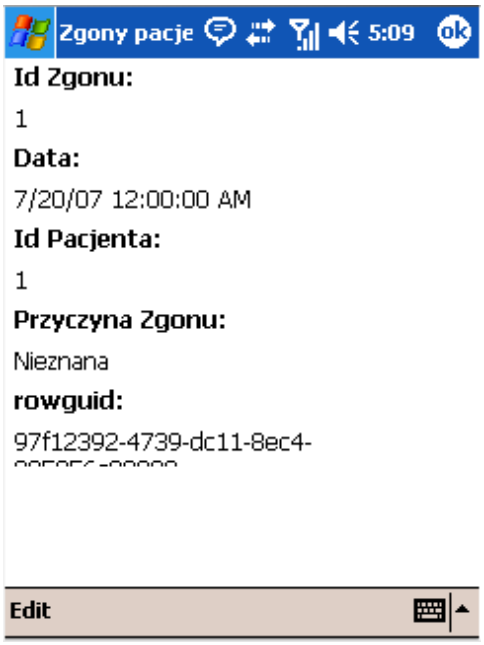


Rys. 32. Zakładka edycji i przeglądania danych historii użytych leków modułu Aesculap H.

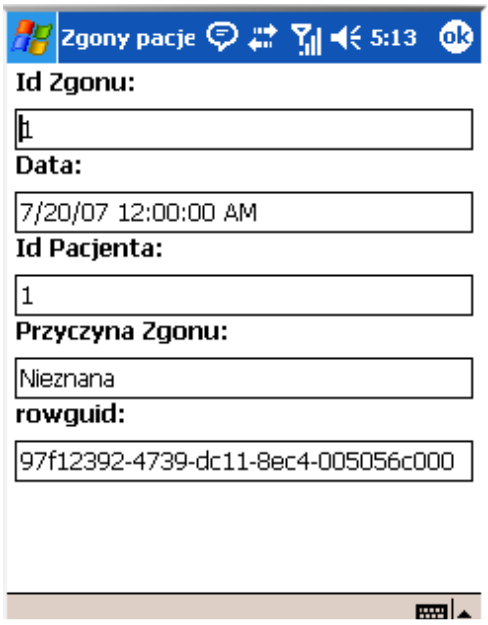


Rys. 33. Zakładka przeglądania grafiku dyżurów lekarskich modułu Aesculap H.

Rys. 34. Przypomnienie o dyżurach modułu Aesculap H.



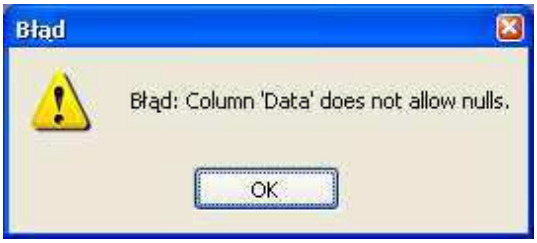
Rys. 35. Podgląd danych modułu Aesculap H.



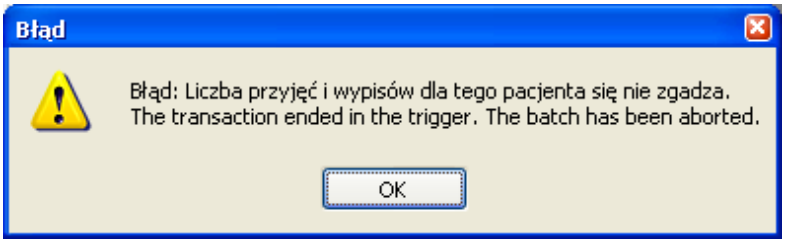
Rys. 36. Edycja danych modułu Aesculap H.

5. Obsługa błędów.

Poniżej przedstawiona zostały wybrane przykłady obsługi błędów w poszczególnych modułach systemu Aesulap. Część z nich związana jest z przedstawionymi wcześniej wyzwalaczami.



Rys. 37. Komunikat o braku wymaganych danych.



Rys. 38. Komunikat wyzwalacza o niezgodności w liczbie przyjęć i wypisów pacjenta.

6. Wymagania systemu.

6.1. Wymagania sprzętowe.

Poszczególne moduły systemu Aesculap zostały przetestowane na różnych platformach testowych i na podstawie tych właśnie testów wyciągnięte zostały wnioski odnośnie wymagań sprzętowych dla każdego z modułów.

- Aesculap R, Aesculap P

Tabela 1. Platformy testowe dla modułów Aesculap R i Aesculap P.

Platforma testowa 1	Platforma testowa 2
Procesor: Intel Pentium M 1.6 GHz	Procesor: Intel Pentium III 450 MHz
RAM: 2048 MB	RAM: 256 MB
Karta graficzna: 96 MB	Karta graficzna: 32 MB
HDD: 60.0 GB	HDD: 40.0 GB

- Aesculap H

Tabela 2. Platformy testowe dla modułu Aesculap H.

Platforma testowa 1	Platforma testowa 2
Procesor: Intel XScale 624 MHz	Procesor: Intel XScale 200 MHz
RAM: 64 MB	RAM: 32 MB
Karta graficzna: Intel 2700 G	Karta graficzna: brak
ROM: 128 MB	ROM: 64 MB

Uruchamianie i działanie poszczególnych aplikacji na przedstawionych powyżej platformach testowych odbywało się płynnie i bez problemów, a zatem za minimalne wymagania sprzętowe można uznać konfiguracje słabszych maszyn spośród użytych do testowania modułów. Urządzenie z procesorem Intel XScale z zegarem 200 MHz jest obecnie jednym z najsłabszych dostępnych obecnie na rynku. W przypadku komputerów stacjonarnych, uruchomienie aplikacji na

jeszcze wolniejszych maszynach nie powinno sprawiać problemów, przy czym maszyna, na której dysku znajdować się będzie baza danych musi spełniać wymagania określone przez firmę Microsoft.

Wymagana ilość wolnego miejsca na dla poszczególnych modułów nie jest duża. Dokładne dane przedstawia poniższa tabela.

Tabela 3. Wymagana ilość wolnego miejsca dla poszczególnych modułów systemu Aesculap.

Moduł	Wymagana ilość wolnego miejsca
Aesculap R	mniej niż 5 MB na HDD
Aesculap P	mniej niż 5 MB na HDD
Aesculap H	około 1 MB ROM

6.2. Wymagane oprogramowanie i ustawienia.

Do poprawnego działania poszczególne moduły systemu Aesculap wymagają obecności określonego oprogramowania, a także konfiguracji niektórych usług . Wymagane oprogramowanie i ustawienia przedstawia poniższa tabela.

Tabela 4. Wymagane oprogramowanie i ustawienia dla poszczególnych modułów systemu Aesculap.

Moduł	Wymagane oprogramowanie i ustawienia
Aesculap R, Aesculap P	1. System operacyjny Windows XP. 2. Zainstalowana baza danych aesculap2 (Microsoft SQL Server 2005). 3. Uruchomiony protokół TCP/IP dla bazy danych. 4. Zainstalowane i skonfigurowane IIS. 5. Skonfigurowana dla bazy danych aesculap2 publikacja typu Merge (koniguracja obejmuje też Web Synchronization).
Aesculap H	1. System operacyjny Windows Mobile 2003. 2. Zainstalowana na komputerze stacjonarnym baza danych (Microsoft SQL Server 2005 Compact Edition). 3. Zainstalowane na komputerze stacjonarnym komponenty Microsoft SQL Server Compact Edition. 4. Skonfigurowana dla bazy aesculap2 (Microsoft SQL Server 2005 Compact Edition) subskrypcja.

6.3. Instalacja i uruchamianie.

Poniższa tabela przedstawia czynności jakie należy wykonać aby zainstalować i uruchomić poszczególne moduły systemu Aesculap.

Tabela 5. Czynności, które należy wykonać aby zainstalować i uruchomić poszczególne moduły systemu Aesculap.

Moduł	Czynności
Aesculap R, Aesculap P	1. Skopiować plik *.exe do wybranej lokalizacji. 2. Uruchomić instalator i postępować zgodnie z jego instrukcjami. 3. Uruchomić aplikację z Menu Start.
Aesculap H	1. Skopiować plik instalacyjny do wybranej lokalizacji przy użyciu programu Microsoft ActiveSync. 2. Uruchomić instalator. 3. Po zakończeniu instalacji w domyślnej lokalizacji, skrót do programu będzie widoczny w menu Start lub w oknie zainstalowanych programów, w zależności od ustawień instalatora.

7. Podsumowanie.

Po zakończeniu prac nad projektem postanowiłem zestawić założenia poczynione przed ich rozpoczęciem z wynikami mojej pracy. W trakcie tworzenia systemu na bieżąco go uzupełniałem i w związku z tym wprowadziłem pewne zmiany, na przykład w sposobie ustalania wizyt i zabiegów. Było to rezultatem porównywania założeń z bieżącą funkcjonalnością systemu. Pomijając drobne problemy z Microsoft Visual Studio 2005 polegające na znikających podczas projektowania interfejsu graficznego kontrolkach, przyjęte rozwiązania okazały się w pełni skuteczne. Dalszy rozwój projektu mógłby iść w kierunku dodawania kolejnych modułów zwiększających jego użyteczność i udoskonalania kierując się sugestiami użytkowników. Podsumowując, tworzenie systemu Aesculap zmobilizowało mnie do poszerzenia mojej wiedzy dotyczącej języka C#, z którym przed rozpoczęciem tworzenia projektu nie miałem do czynienia i było ciekawym doświadczeniem, gdyż nie zawsze istnieje okazja stworzenia projektu w całości samodzielnie.

8. Literatura.

Internet:

- msdn.com
- w3schools.com

SPIS TREŚCI