SZKOŁA WYŻSZA im. PAWŁA WŁODKOWICA
w PŁOCKU

WYDZIAŁ INFORMATYKI

API dla aplikacji mobilnej dla klienta kliniki weterynaryjnej

Programowanie interfejsów (sem VI)

Przygotował Kamil Zima

Spis teści

1. Geneza wyboru tematu

Rynek polskich programów zarządania kliniką weterynaryjną weterynaryjną posiada szereg rozwiązań w różnych kategoriach cenowych. Większość programów w przystępnej cenie nie posiada jeszcze możliwości tworznie JPK_MAG oraz nie posiada żadnych funkcjonalności zbliżających klienta do kliniki (takich jak aplikacja mobilna lub powiadomienia SMS), aplikacje które spełniają większość wymagań klinik weterynaryjnych są duże, niekoniecznie przejrzyste i posiadają zawiłe reguły płatności (płatności ukryte, konieczność opłaty aktualizacji za miesiące w których nie korzystało się z programu lub wsparcia.)

2. Charakterystyka programu

Po przeanalizaowanu wymagań okazuje się, że odpowidnią komunikację zdalną z klientem zapewni aplikacja mobilna, ma ona umożliwić klientowi po zalogowaniu się przeglądanie książeczki zdrowia swoich zwierząt, uzupełnienie swoich spostrzeżeń na temat zdrowia lub zachowania do wglądu lekarza oraz wgląd w terminarz który odpowiednio wcześniej przypominałby o zaplanowanej wizycie. Będzie można również za jej pomocą sprawdzić dokładny adres przychodni weterenaryjnej, godziny otwarcia czy specjalizację personelu oraz ewentualne osiągnięcia, ma to zwiększyć lojalność klienta przez zmniejszenie ryzyka natknięcia się na konkuręcyjną ofertę w internecie przy wyszukiwaniu tych informacji, a także kształtować wizerunek nowoczesnej i korzystającej z technologii kliniki.

Serwerowa część oprogramowania (API) wykonana jest w technolgii .NET korzystając z bazy MSQL; aplikacje mobile przeznaczone są na system Android. Dane pomiędzy aplikacją serwerową, a mobilną przesyłane są w formacie JSON.

3. Analiza wymagań

3.1 Grupa docelowa

Klienci kliniki weterenaryjnej korzystającej z systemu do zarządzania kliniką

3.2 Wymagania funkcjonalne

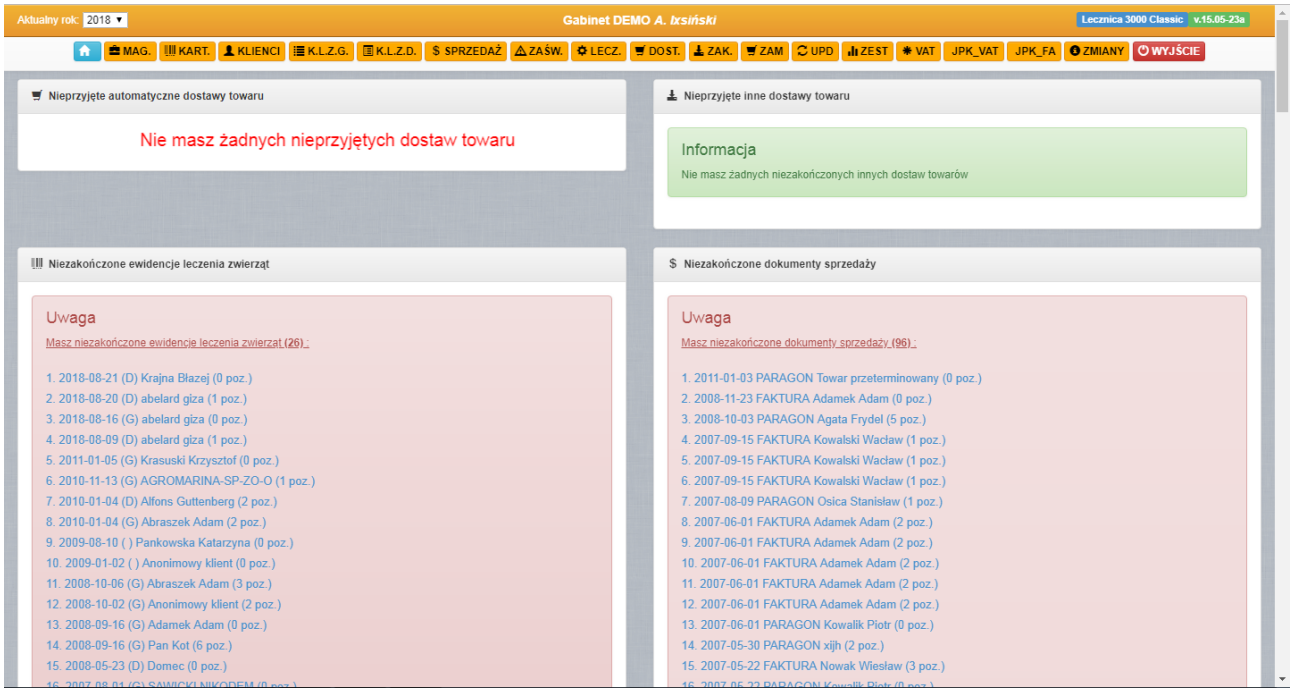
- Przeglądania karty zdrowia zwierzęcia
- Terminarz wraz z funkcjonalnością powiadomienia przed wizytą
- Wysłanie prośby o umówienie wizyty
- Uaktualnienie swoich danych kontaktowych
- Wyświetlanie podstawowych danych kliniki oraz jej pracowników

3.3 Wymagania niefunkcjonalne

- Dostępność dla jak najwiekszej liczby użytkowników (wybór platformy Android)
- Łatwa instalacja

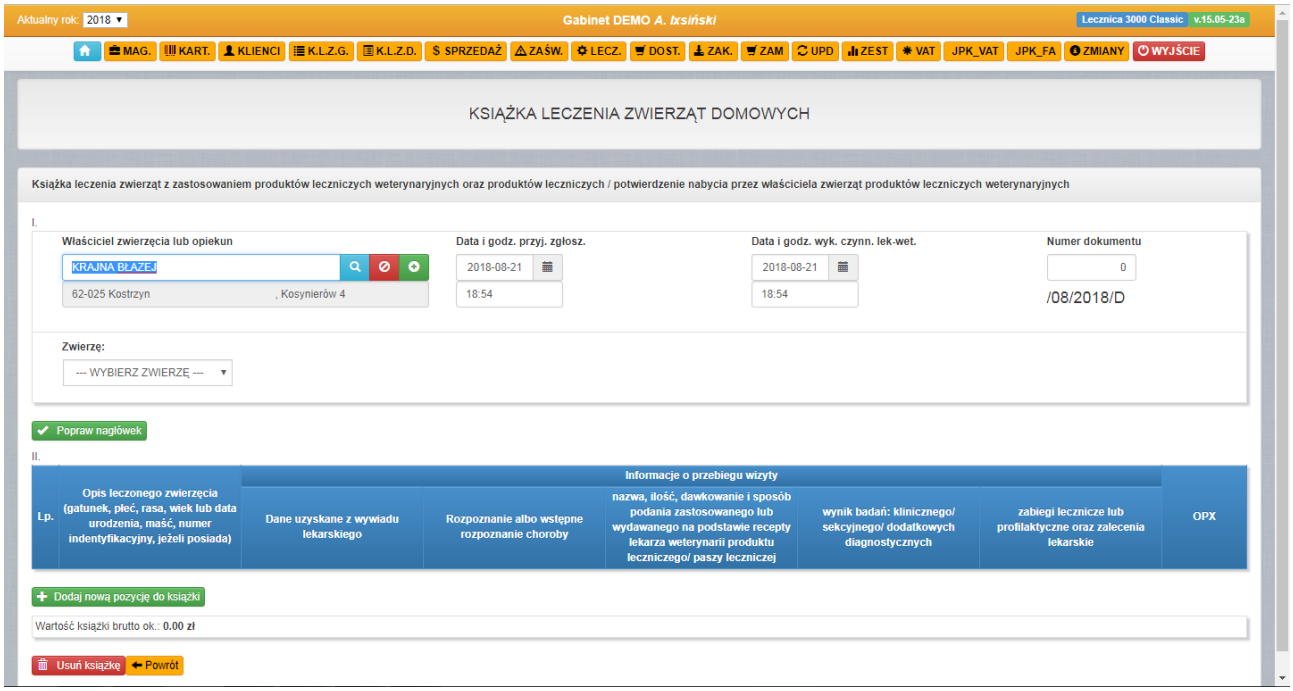
4. Przegląd i analiza rozwiązań konkuencyjnych

4.1.1 Lecznica-3000 Classic

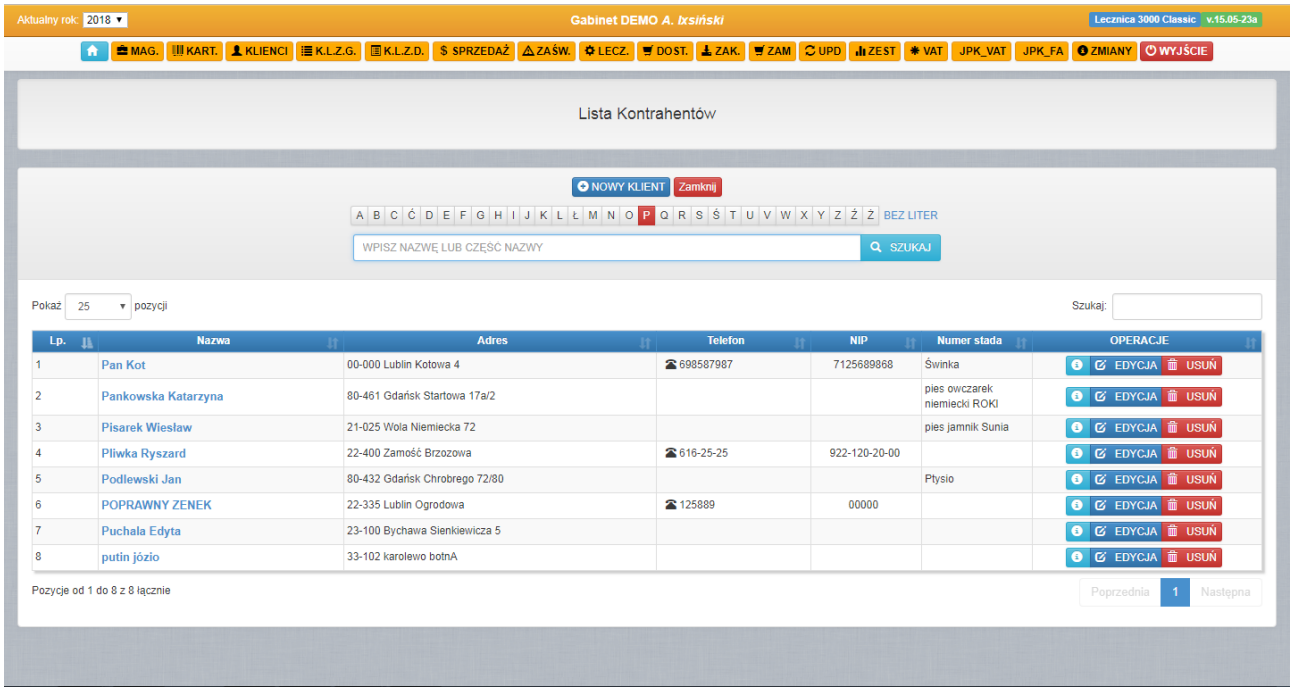


Ilustracja 1: Lecznica 3000 strona startowa

Na screenie powyżej mamy stronę startową programu, wyświetlane są na niej rzeczy pilne, wymagające uwagi użytkownika, głównie dotyczące dokumentów oraz towarów.



Ilustracja 2: Lecznica 3000 moduł Książeczka Leczenia Zwierząt Domowych



Ilustracja 3: Lecznica 3000 moduł Lista Kontrahentów

Jak widać program jest bardzo prosty i oszczędny w formie, nie wygląda nowocześnie, a jego interface nie jest zbyt intuicyjny. Brak mu przejrzystości. Nie posiada modułu kalendarza, który w jasny i czytelny sposób pokazywałby choćby listę pacjentów umówionych na dany dzień. Nie posiada wersji mobilnej ani dla kliniki ani dla właścicieli leczonych tam zwierząt.

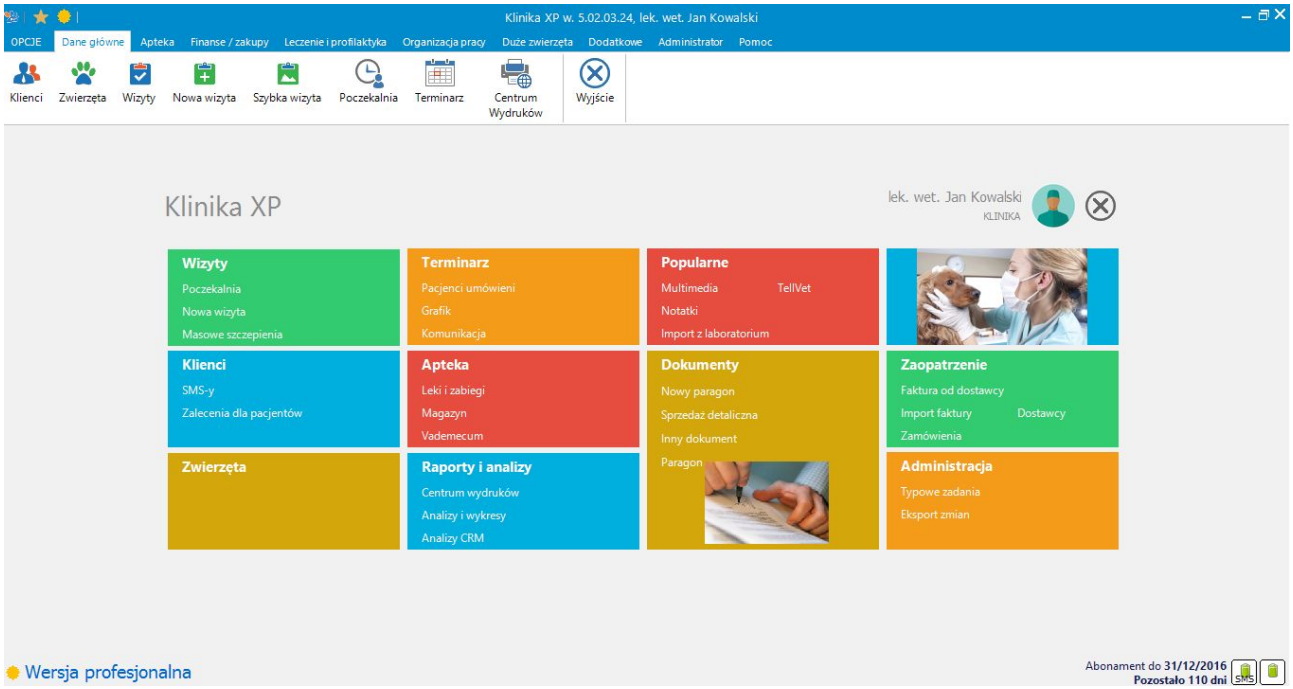
Zalety:

- Przystępna cena.
- Posiada podstawowe moduły niezbędne do zarządzania kliniką.

Wady:

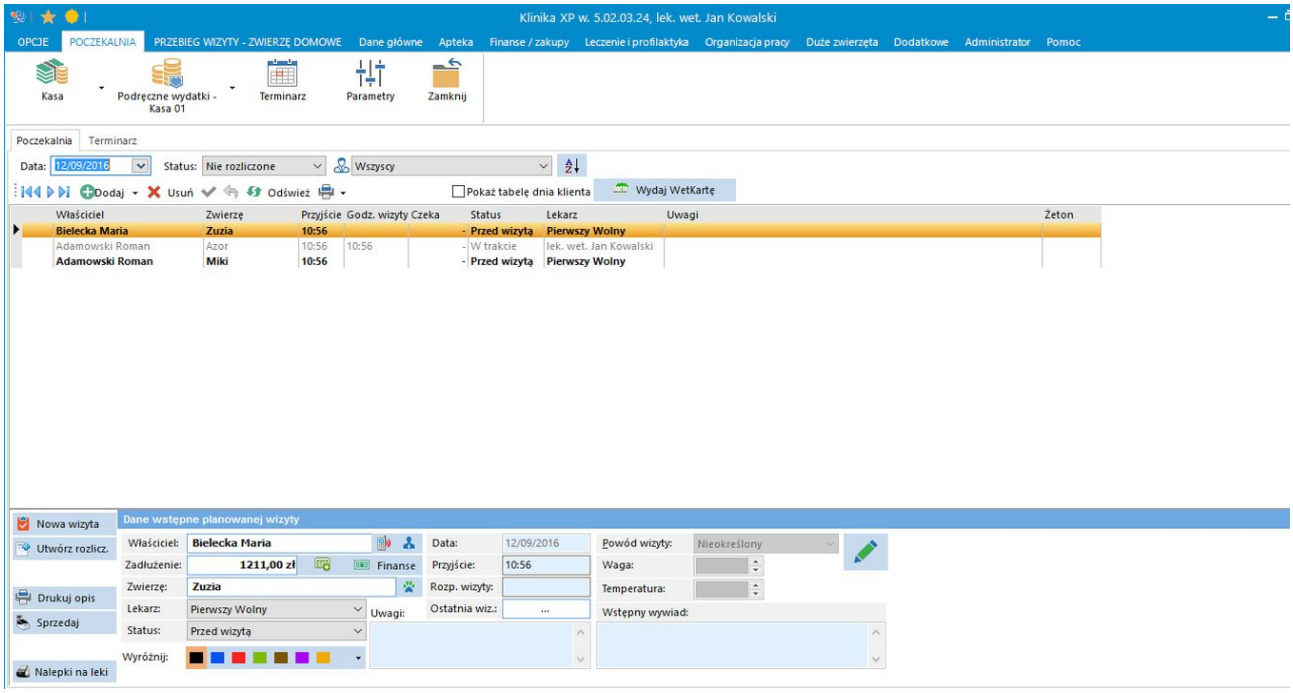
- Przestarzały, nieczytelny i nieintuicyjny interface.
- Brak możliwości obsługi mobilnej.
- Brak możliwości kontaktu z pacjentem, wysyłania powiadomień i przypomnień.
- Brak możliwości umawiania wizyt online.
- Brak czytelnego terminarza.
- Brak możliwości doboru modułów.

4.1.2 KlinikaXP

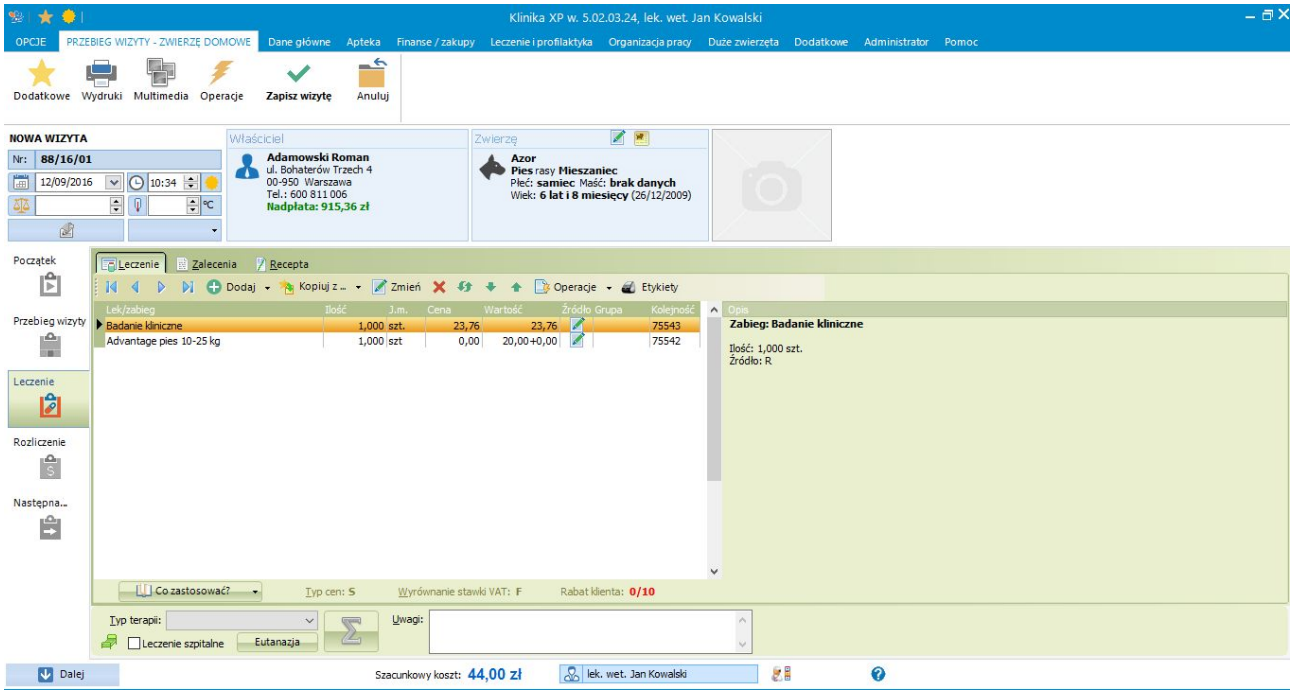


Ilustracja 4: KlinikaXP strona startowa

Na screenie powyżej mamy stronę startową programu, jak widać wygląda on bardziej nowocześnie i czytelnie od poprzedniego i zawiera więcej przydatnych zakładek.



Ilustracja 5: KlinikaXP moduł Poczekalnia



Ilustracja 6: KlinikaXP moduł Przebieg Wizyty

Na pierwszy rzut oka, program wydaje się być przejrzysty i intuicyjny, niestety kiedy bardziej zagłębimy się w niego i dotrzemy choćby do modułu przebiegu wizyty, zauważamy iż staje się nieczytelny i chaotyczny przez nagromadzenie okienek przeróżnych kolorów i rozmiarów. Brak mu konsekwentnie wykonanego, spójnego interface'u.

W prawdzie posiada terminarz, ale aplikacji dla klienta.

Dodatkowo zniechęca cennik zamieszczony na stronie programu, w którym dowiadujemy się, że sama licencja na wersję podstawową jest kosztowna i dodatkowo za każdą aktualizację, moduł czy dodatkowe stanowisko trzeba słono zapłacić.

Zalety:

- Możliwość doboru modułów.
- Wersja w chmurze.
- Terminarz.

Wady:

- Cena.
- Chaotyczny interface.
- Brak możliwości kontaktu z pacjentem, wysyłania powiadomień i przypomnień.
- Brak możliwości umawiania wizyt online.

5. Charakterystyka użytkownika

5.1 Klient kliniki

Osoba zarejestrowana w klinice weterenaryjnej posiadająca telefon z systemem Android.

6. Zadania realizowane przez użytkownika

- Logowanie
- Zmiana hssła
- Wyświetlenie listy zwierząt
- Dodanie opisu zwierzęcia
- Wyświetlenie książeczki zdrowia zwierzęcia
- Wyświetlenie informacji o swojej klinice
- Wyświetlenie informacji o lekarzach
- Wyświetlenie terminarza

7. Diagramy

7.1 Diagram przypaków użycia

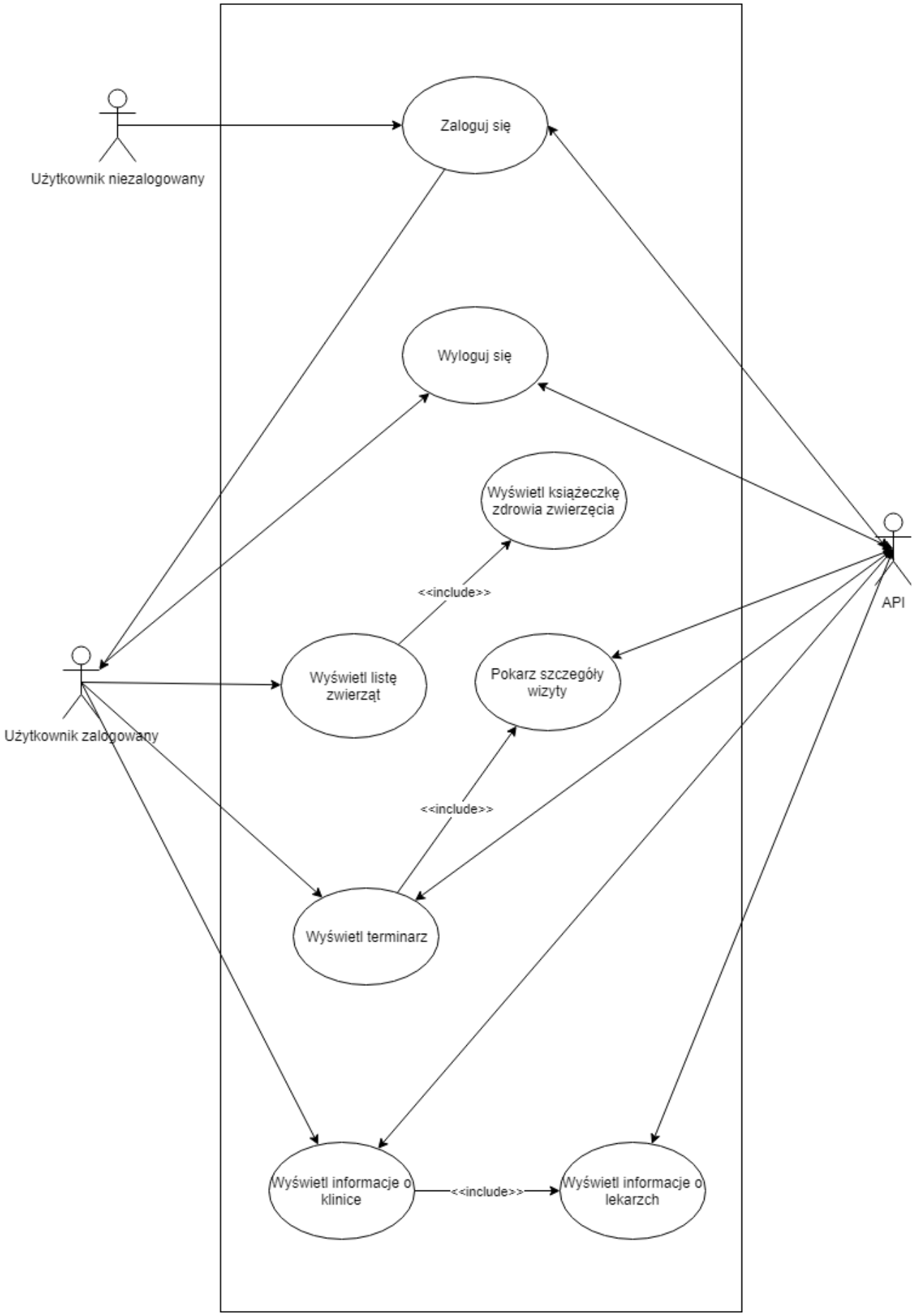


Diagram 1: Przypadki użycia

7.2 Diagram ERD

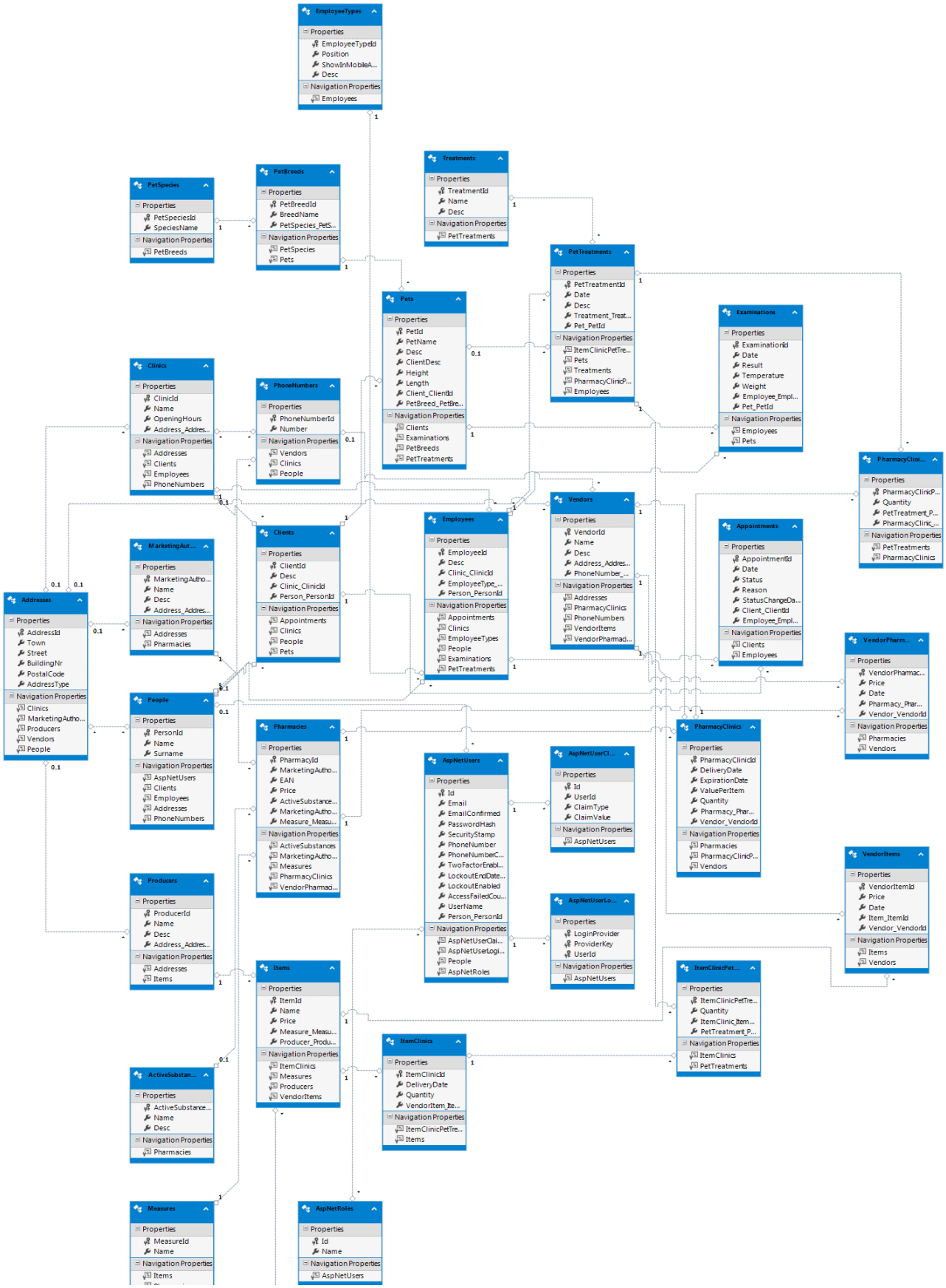


Diagram 2: ERD

7.3 Diagram nawigacji

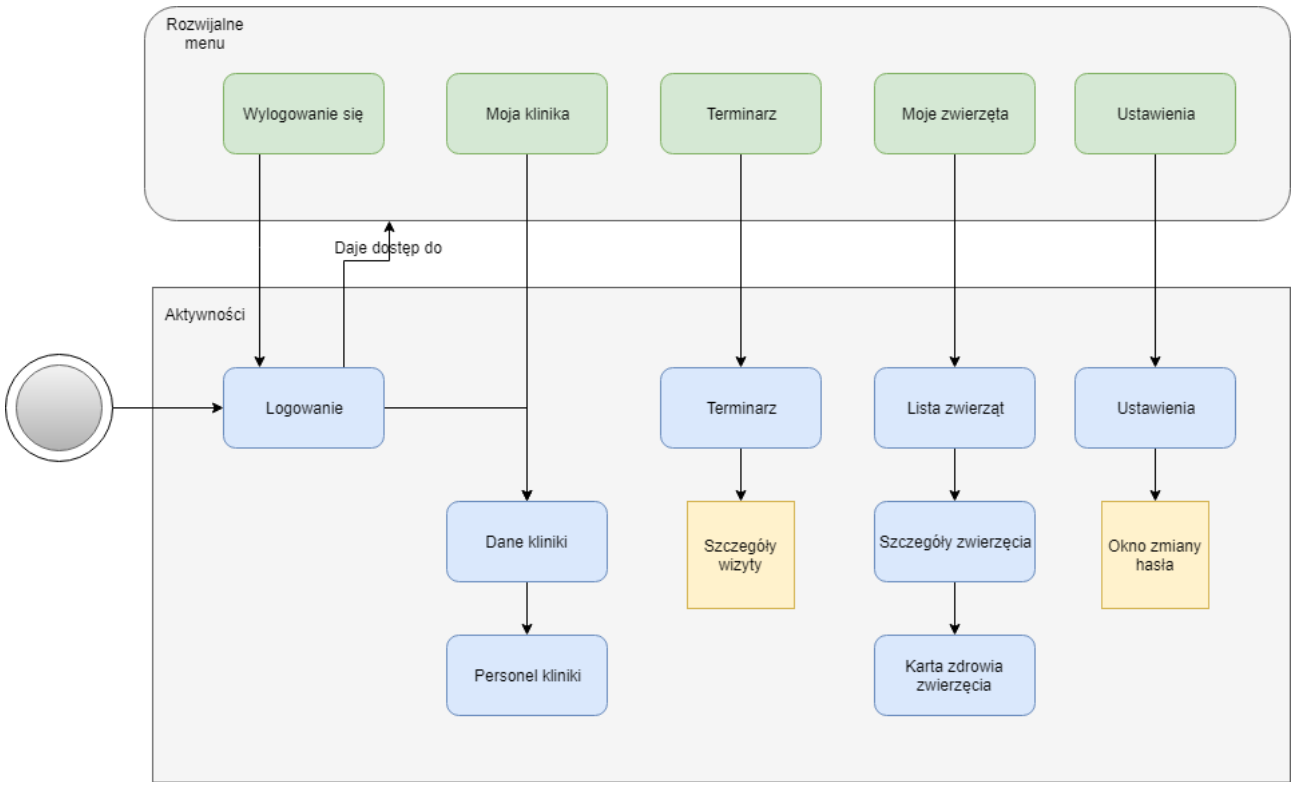


Diagram 3: Nawigacja

8. Rozwiązania techniczne

8.1 Główne modele

Aplikacja została napisana z podejściem "Code First", gdzie Entity Framework tworzy na podstawie modeli(Entity) odpowiednie migracje, które przekładają się na tabele w bazie danych.

Oto najważniejsze z nich niezbędne do właściwej pracy API.

```
7   namespace KlinikWetApi.Models
8   {
9       public class Clinic
10      {
11          public Clinic()
12          {
13              ClinicPhoneNumber = new List<PhoneNumber>();
14          }
15          [Key]
16          public int ClinicId { get; set; }
17          public Address Address { get; set; }
18          [Required]
19          [StringLength(255)]
20          public string Name { get; set; }
21          [Required]
22          [StringLength(255)]
23          public string OpeningHours { get; set; }
24          public virtual ICollection<PhoneNumber> ClinicPhoneNumber { get; set; }
25      }
26  }
```

Kod 1: Klasa Clinic

```
7   namespace KlinikWetApi.Models
8   {
9       public class Employee
10      {
11          public Employee()
12          {
13              EmployeePetTreatment = new List<PetTreatment>();
14          }
15
16          [Key]
17          public int EmployeeId { get; set; }
18          [Required]
19          public Person Person { get; set; }
20          [Required]
21          public EmployeeType EmployeeType { get; set; }
22          [Required]
23          public Clinic Clinic { get; set; }
24          [StringLength(255)]
25          public string Desc { get; set; }
26
27          public virtual ICollection<PetTreatment> EmployeePetTreatment { get; set; }
28      }
29  }
```

Kod 2: Klasa Employee

```
7   namespace KlinikWetApi.Models
8   {
9       public class Person
10      {
11          public Person()
12          {
13              PersonPhoneNumber = new List<PhoneNumber>();
14              PersonAddress = new List<Address>();
15              PersonAspNetUsers = new List<AspNetUsers>();
16          }
17          [Key]
18          public int PersonId { get; set; }
19          [Required]
20          [StringLength(127)]
21          public string Name { get; set; }
22          [Required]
23          [StringLength(127)]
24          public string Surname { get; set; }
25          public virtual ICollection<PhoneNumber> PersonPhoneNumber { get; set; }
26          public virtual ICollection<Address> PersonAddress { get; set; }
27          public virtual ICollection<AspNetUsers> PersonAspNetUsers { get; set; }
28      }
29  }
```

Kod 3: Klasa Person

```
7   namespace KlinikWetApi.Models
8   {
9       public enum Sex { male = 0, female }
10      public class Pet
11      {
12          public Pet()
13          {
14              PetTreatment = new List<PetTreatment>();
15          }
16          [Key]
17          public int PetId { get; set; }
18          [Required]
19          public Client Client { get; set; }
20          [Required]
21          public PetBreed PetBreed { get; set; }
22          [Required]
23          [StringLength(63)]
24          public string PetName { get; set; }
25          [StringLength(511)]
26          public string Desc { get; set; }
27          [StringLength(511)]
28          public string ClientDesc { get; set; }
29          public DateTime BirthDate { get; set; }
30          public Sex Sex { get; set; }
31          public Decimal Height { get; set; }
32          public Decimal Length { get; set; }
33
34          public virtual ICollection<PetTreatment> PetTreatment { get; set; }
35
36      }
37  }
```

Kod 4: Klasa Pet

```
7   namespace KlinikWetApi.Models
8   {
9       public class PhoneNumber
10      {
11          public PhoneNumber()
12          {
13              PersonPhoneNumber = new List<Person>();
14              ClinicPhoneNumber = new List<Clinic>();
15          }
16          [Key]
17          public int PhoneNumberId { get; set; }
18          [Required]
19          [StringLength(15)]
20          public string Number { get; set; }
21          public virtual ICollection<Person> PersonPhoneNumber { get; set; }
22          public virtual ICollection<Clinic> ClinicPhoneNumber { get; set; }
23      }
24  }
```

Kod 5: Klasa PhoneNumber

```
7   namespace KlinikWetApi.Models
8   {
9       public class PetTreatment
10      {
11          public PetTreatment()
12          {
13              EmployeePetTreatment = new List<Employee>();
14          }
15          [Key]
16          public int PetTreatmentId { get; set; }
17          [Required]
18          public DateTime Date { get; set; }
19          [Required]
20          public Treatment Treatment { get; set; }
21          [Required]
22          public Pet Pet { get; set; }
23          [StringLength(255)]
24          public string Desc { get; set; }
25
26          public virtual ICollection<Employee> EmployeePetTreatment { get; set; }
27      }
28  }
```

Kod 6: Klasa PetTreatment

```
7   namespace KlinikWetApi.Models
8   {
9       public class Treatment
10      {
11          [Key]
12          public int TreatmentId { get; set; }
13          [Required]
14          [StringLength(255)]
15          public string Name { get; set; }
16          [StringLength(511)]
17          public string Desc { get; set; }
18      }
19  }
```

Kod 7: Klasa Treatment

```
7   namespace KlinikWetApi.Models
8   {
9       public class Examination
10      {
11          [Key]
12          public int ExaminationId { get; set; }
13          [Required]
14          public DateTime Date { get; set; }
15          [Required]
16          public Employee Employee { get; set; }
17          [Required]
18          public Pet Pet { get; set; }
19          [Required]
20          [StringLength(255)]
21          public string Result { get; set; }
22          public Decimal Temperature { get; set; }
23          public Decimal Weight { get; set; }
24      }
25  }
```

Kod 8: Klasa Examination

8.2 ModelView

Klasy modeli bazy danych posiadają wiele nadmiarowych danych z punktu widzenia aplikacji mobilnej, w związku z tym jako odpowiedź na zapytania wysyła się dane w klasach "ViewModel" pozbawionych zbędnych informacji.

```
6   namespace KlinikWetApi.Models.ViewModels
7   {
8       public class AddressViewModel
9       {
10          public int AddressId { get; set; }
11          public string Town { get; set; }
12          public string Street { get; set; }
13          public string BuildingNr { get; set; }
14          public string PostalCode { get; set; }
15          public string AddressType { get; set; }
16      }
17  }
```

Kod 9: Klasa AddressViewModel

```
6   namespace KlinikWetApi.Models.ViewModels
7   {
8       public class AppointmentPetViewModel
9       {
10          public int PetId { get; set; }
11          public string Name { get; set; }
12      }
13  }
```

Kod 10: Klasa AppointmentViewModel

```
6   namespace KlinikWetApi.Models.ViewModels
7   {
8       public class AppointmentModelView
9       {
10          public int AppointmentId { get; set; }
11          public DateTime Date { get; set; }
12          public EmployeeViewModel Employee { get; set; }
13          public AppointmentStatus Status { get; set; }
14          public DateTime StatusChangeDate { get; set; }
15          public string ClientUserName { get; set; }
16      }
17  }
```

Kod 12: AppointmentModelView

```
6   namespace KlinikWetApi.Models.ViewModels
7   {
8       public class PhoneNumberViewModel
9       {
10          public int PhoneNumberId { get; set; }
11          public int ClinicId { get; set; }
12          public string Number { get; set; }
13      }
14  }
```

Kod 11: Klasa PhoneNumerViewModel

```
6   namespace KlinikWetApi.Models.ViewModels
7   {
8       public class PetTreatmentViewModel
9       {
10          public int PetTreatmentId { get; set; }
11          public DateTime Date { get; set; }
12          public int PetId { get; set; }
13          public TreatmentViewModel Treatment { get; set; }
14          public string PetTreatmentDesc { get; set; }
15          public EmployeeViewModel EmployeePetTreatment { get; set; }
16          public string ClientUserName { get; set; }
17      }
18  }
```

Kod 13: Klasa PetTreatmentViewModel

```
6   namespace KlinikWetApi.Models.ViewModels
7   {
8       public class TreatmentViewModel
9       {
10          public int TreatmentId { get; set; }
11          public string TreatmentName { get; set; }
12      }
13  }
```

Kod 14: Klasa TreatmentViewModel

```
6   namespace KlinikWetApi.Models.ViewModels
7   {
8       public class PetViewModel
9       {
10          public int PetId { get; set; }
11          public string PetSpecies { get; set; }
12          public string PetBreed { get; set; }
13          public string PetName { get; set; }
14          public string ClientDesc { get; set; }
15          public Decimal Height { get; set; }
16          public Decimal Weight { get; set; }
17          public Decimal Length { get; set; }
18          public Sex Sex { get; set; }
19          public DateTime BirthDate { get; set; }
20          public string ClientUserName { get; set; }
21      }
22  }
```

Kod 15: Klasa PetViewModel

```
6   namespace KlinikWetApi.Models.ViewModels
7   {
8       public class EmployeeViewModel
9       {
10          public int EmployeeId { get; set; }
11          public int ClinicId { get; set; }
12          public bool ShowInMobileApp { get; set; }
13          public string EmployeeName { get; set; }
14          public string Surname { get; set; }
15          public string Position { get; set; }
16          public string EmployeeDesc { get; set; }
17      }
18  }
```

Kod 16: Klasa EmployeeViewModel

8.3 API

Zapytania Api i przykładowe odpowiedzi lub parametry:

8.3.1 POST api/Accounts/Token

Zapytanie o token autoryzujący niezbędny do wykonania reszty zapytań.

Przykładowe parametry:


```
{
  "Username": "sample string 1",
  "Grant_token": "password",
  "Password": "sample string 2"
}
```


8.3.2 POST api/Account/ChangePassword

Zmiana hasła. Wymaga tokenu. Przykładowe parametry.


```
{
  "OldPassword": "sample string 1",
  "NewPassword": "sample string 2",
  "ConfirmPassword": "sample string 3"
}
```


8.3.3 GET api/Clients

Usyskanie własnych danych klienta. Wymaga tokenu. Przykładowa odpowiedź.


```
{
  "PersonId": 1,
  "ClinicId": 2,
  "Email": "sample string 3",
  "ClientName": "sample string 4",
  "ClientSurname": "sample string 5"
}
```


8.3.4 GET api/Pets

Usyskanie listy zwierząt. Wymaga tokenu. Przykładowa odpowiedź.


```
[
  {
    "PetId": 1,
```

```
    "PetSpecies": "sample string 2",
    "PetBreed": "sample string 3",
    "PetName": "sample string 4",
    "ClientDesc": "sample string 5",
    "Height": 6.0,
    "Weight": 7.0,
    "Length": 8.0,
    "Sex": 0,
    "BirthDate": "2018-09-07T22:23:45.0089448+00:00",
    "ClientUserName": "sample string 10"
  },
  {
    "PetId": 1,
    "PetSpecies": "sample string 2",
    "PetBreed": "sample string 3",
    "PetName": "sample string 4",
    "ClientDesc": "sample string 5",
    "Height": 6.0,
    "Weight": 7.0,
    "Length": 8.0,
    "Sex": 0,
    "BirthDate": "2018-09-07T22:23:45.0089448+00:00",
    "ClientUserName": "sample string 10"
  }
]
```


8.3.5 PUT api/Pets/{id}?petDesc={petDesc}

Usyskanie listy umówionych spotkań. Wymaga tokenu. Przykładowa odpowiedź.

GET api/Appointments


```
[
  {
    "AppointmentId": 1,
    "Date": "2018-09-07T22:24:44.6199267+00:00",
    "Employee": {
      "EmployeeId": 1,
```

```
    "ClinicId": 2,
    "ShowInMobileApp": true,
    "EmployeeName": "sample string 4",
    "Surname": "sample string 5",
    "Position": "sample string 6",
    "EmployeeDesc": "sample string 7"
   },
   "Status": 1,
   "StatusChangeDate": "2018-09-07T22:24:44.6199267+00:00",
   "ClientUserName": "sample string 4"
  },
  {
   "AppointmentId": 1,
   "Date": "2018-09-07T22:24:44.6199267+00:00",
   "Employee": {
    "EmployeeId": 1,
    "ClinicId": 2,
    "ShowInMobileApp": true,
    "EmployeeName": "sample string 4",
    "Surname": "sample string 5",
    "Position": "sample string 6",
    "EmployeeDesc": "sample string 7"
   },
   "Status": 1,
   "StatusChangeDate": "2018-09-07T22:24:44.6199267+00:00",
   "ClientUserName": "sample string 4"
  }
]
```


8.3.6 GET api/Clinics/{ClinicId}

Usyskanie danych kliniki. Niewymaga tokenu. Przykładowa odpowiedź.


```
{
  "ClinicId": 1,
  "Address": {
   "AddressId": 1,
   "Town": "sample string 2",
```

```
    "Street": "sample string 3",
    "BuildingNr": "sample string 4",
    "PostalCode": "sample string 5",
    "AddressType": "sample string 6"
  },
  "ClinicName": "sample string 2",
  "OpeningHours": "sample string 3",
  "ClinicPhoneNumber": [
    {
      "PhoneNumberId": 1,
      "ClinicId": 2,
      "Number": "sample string 3"
    },
    {
      "PhoneNumberId": 1,
      "ClinicId": 2,
      "Number": "sample string 3"
    }
  ]
}
```


8.3.7 GET api/PetTreatments/{PetId}

Usyskanie listy zabiegów wybranego zwierzęcia. Wymaga tokenu. Przykładowa odpowiedź.


```
{
  "PetTreatmentId": 1,
  "Date": "2018-09-07T22:25:32.2820985+00:00",
  "PetId": 3,
  "Treatment": {
    "TreatmentId": 1,
    "TreatmentName": "sample string 2"
  },
  "PetTreatmentDesc": "sample string 4",
  "EmployeePetTreatment": {
    "EmployeeId": 1,
    "ClinicId": 2,
    "ShowInMobileApp": true,
```

```
    "EmployeeName": "sample string 4",
    "Surname": "sample string 5",
    "Position": "sample string 6",
    "EmployeeDesc": "sample string 7"
  },
  "ClientUserName": "sample string 5"
}
```


GET api/Employees/{ClinicId}


```
[
  {
    "EmployeeId": 1,
    "ClinicId": 2,
    "ShowInMobileApp": true,
    "EmployeeName": "sample string 4",
    "Surname": "sample string 5",
    "Position": "sample string 6",
    "EmployeeDesc": "sample string 7"
  },
  {
    "EmployeeId": 1,
    "ClinicId": 2,
    "ShowInMobileApp": true,
    "EmployeeName": "sample string 4",
    "Surname": "sample string 5",
    "Position": "sample string 6",
    "EmployeeDesc": "sample string 7"
  }
]
```

8.4 Kontroler aplikacji mobilnej

Kod obsługujący wcześniej podane zapytania api.

```
15      public RestResponse postLogin(String mEmail, String mPassword) {
16          try {
17              apiConnection.setRequest("/api/token");
18              apiConnection.setReqMethod("POST");
19              apiConnection.addBodyParameter( parameter: "grant_type",  value: "password");
20              apiConnection.addBodyParameter( parameter: "username", mEmail);
21              apiConnection.addBodyParameter( parameter: "password", mPassword);
22              RestResponse response = apiConnection.execute();
23              return response;
24          } catch (Exception e) {
25              e.getMessage();
26          }
27          return exceptionRequestResponse;
28      }
29
30      public RestResponse getClinic(String clinicId) {
31          try {
32              apiConnection.setRequest("/api/clinics/{clinicId}");
33              apiConnection.setReqMethod("GET");
34              apiConnection.addUrlSegment( parameter: "{clinicId}", clinicId);
35              RestResponse response = apiConnection.execute();
36              return response;
37          } catch (Exception e) {
38              e.getMessage();
39          }
40          return exceptionRequestResponse;
41      }
42
43      public RestResponse getPets() {
44          try {
45              apiConnection.setRequest("/api/Pets");
46              apiConnection.setReqMethod("GET");
47              apiConnection.setAuthorize(true);
48              RestResponse response = apiConnection.execute();
49              return response;
50          } catch (Exception e) {
51              e.getMessage();
52          }
53          return exceptionRequestResponse;
54      }
```

Kod 17: Kontroler cz.1

```
56      public RestResponse getClient() {
57          try {
58              apiConnection.setRequest("/api/Clients");
59              apiConnection.setReqMethod("GET");
60              apiConnection.setAuthorize(true);
61              RestResponse response = apiConnection.execute();
62              return response;
63          } catch (Exception e) {
64              e.getMessage();
65          }
66          return exceptionRequestResponse;
67      }
68
69      public boolean putPet(String PetId, String PetDesc) {
70          try {
71              apiConnection.setRequest("/api/Pets/{PetId}");
72              apiConnection.setReqMethod("PUT");
73              apiConnection.setAuthorize(true);
74              apiConnection.addUrlSegment( parameter: "{PetId}", PetId);
75              apiConnection.addBodyParameter( parameter: "PetDesc", PetDesc);
76              RestResponse response = apiConnection.execute();
77              if (response.getResponseCode() == 204)
78                  return true;
79          } catch (Exception e) {
80              e.getMessage();
81          }
82          return false;
83      }
84
85      public RestResponse getPetTreatment(String PetId) {
86          try {
87              apiConnection.setRequest("/api/PetTreatments/{PetId}");
88              apiConnection.setReqMethod("GET");
89              apiConnection.setAuthorize(true);
90              apiConnection.addUrlSegment( parameter: "{PetId}", PetId);
91              RestResponse response = apiConnection.execute();
92              return response;
93          } catch (Exception e) {
94              e.getMessage();
95          }
96          return exceptionRequestResponse;
97      }
```

Kod 18: Kontroler cz.2

```
99      public RestResponse getAppointments() {
100         try {
101             apiConnection.setRequest("/api/Appointments");
102             apiConnection.setReqMethod("GET");
103             apiConnection.setAuthorize(true);
104             RestResponse response = apiConnection.execute();
105             return response;
106         } catch (Exception e) {
107             e.getMessage();
108         }
109         return exceptionRequestResponse;
110     }
111
112     public RestResponse getEmployees(String ClinicId) {
113         try {
114             apiConnection.setRequest("/api/Employees/{ClinicId}");
115             apiConnection.setReqMethod("GET");
116             apiConnection.setAuthorize(true);
117             apiConnection.addUrlSegment( parameter: "{ClinicId}", ClinicId);
118             RestResponse response = apiConnection.execute();
119             return response;
120         } catch (Exception e) {
121             e.getMessage();
122         }
123         return exceptionRequestResponse;
124     }
125
126     public void postLogout() {
127         try {
128             apiConnection.setRequest("api/Account/Logout");
129             apiConnection.setReqMethod("POST");
130             apiConnection.setAuthorize(true);
131             RestResponse response = apiConnection.execute();
132             Token.reset();
133         } catch (Exception e) {
134             e.getMessage();
135         }
136     }
```

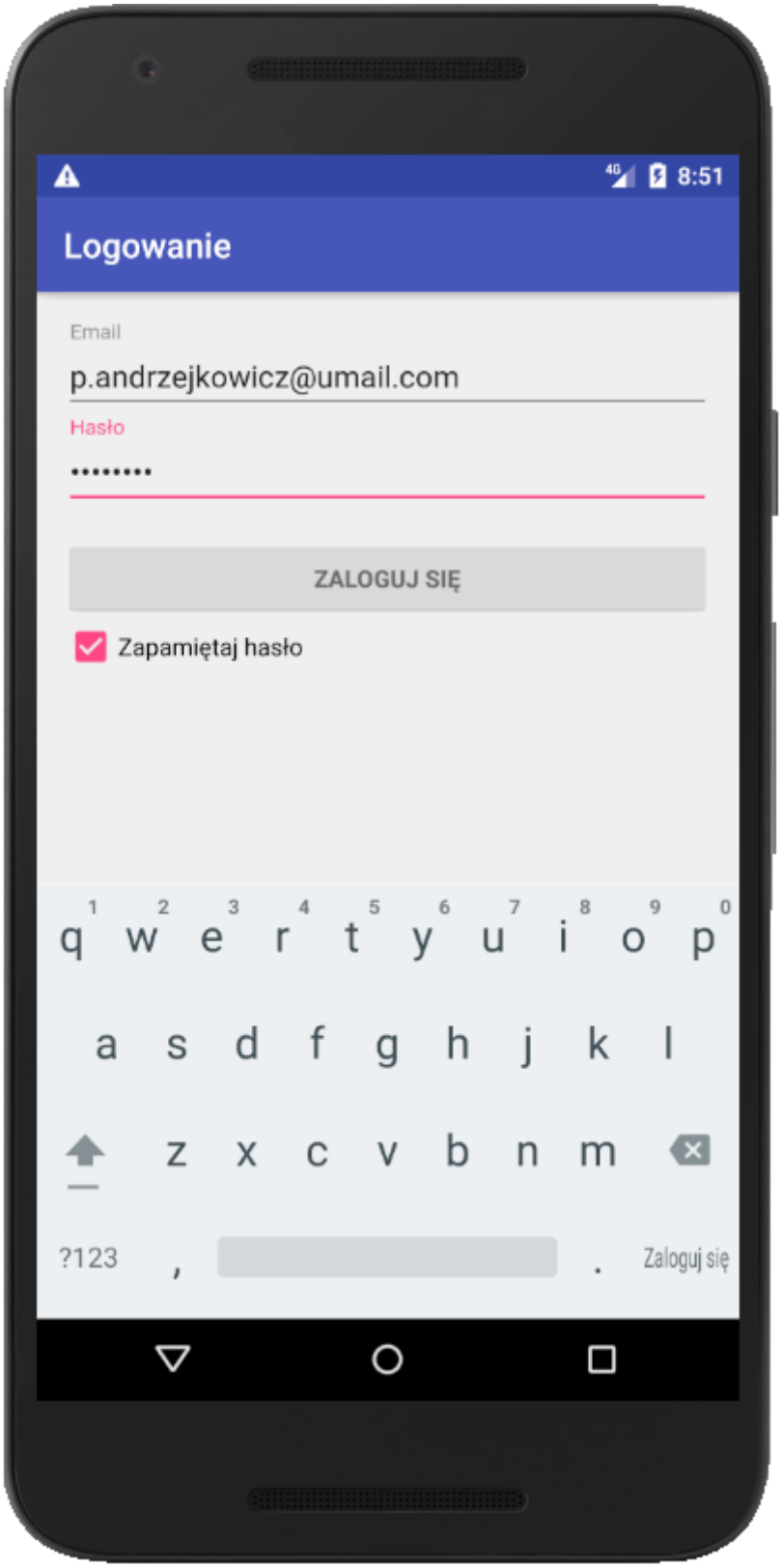
Kod 19: Kontroler cz.3

```
138     public RestResponse postChangePassword(String OldPassword, String NewPassword, String ConfirmPassword) {
139         try {
140             apiConnection.setRequest("/api/Account/ChangePassword");
141             apiConnection.setReqMethod("POST");
142             apiConnection.setAuthorize(true);
143             apiConnection.addBodyParameter( parameter: "OldPassword", OldPassword);
144             apiConnection.addBodyParameter( parameter: "NewPassword", NewPassword);
145             apiConnection.addBodyParameter( parameter: "ConfirmPassword", ConfirmPassword);
146             RestResponse response = apiConnection.execute();
147             return response;
148         } catch (Exception e) {
149             e.getMessage();
150         }
151         return exceptionRequestResponse;
152     }
153 }
```

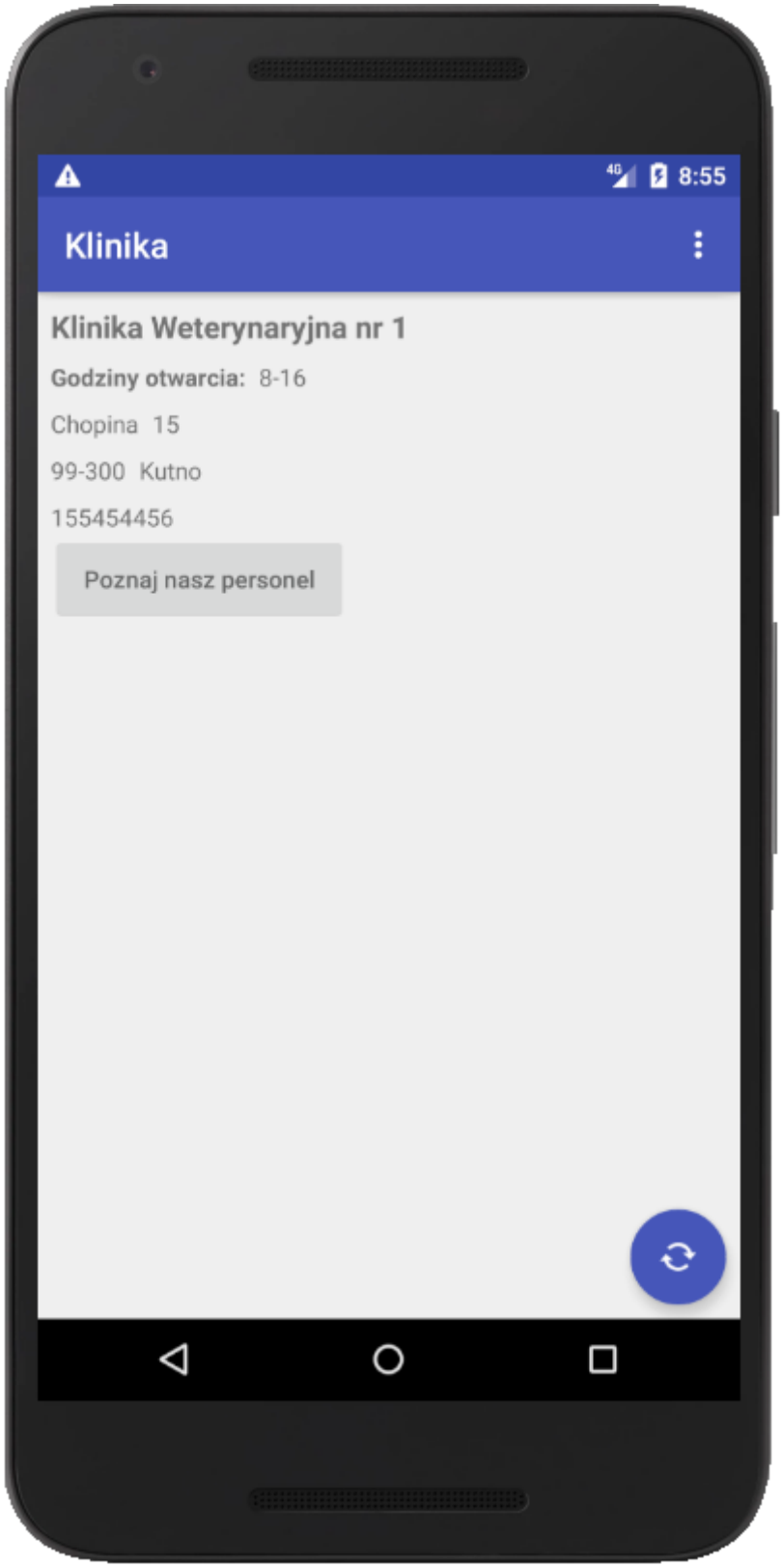
Kod 20: Kontroler cz.4

9. Widoki aplikacji

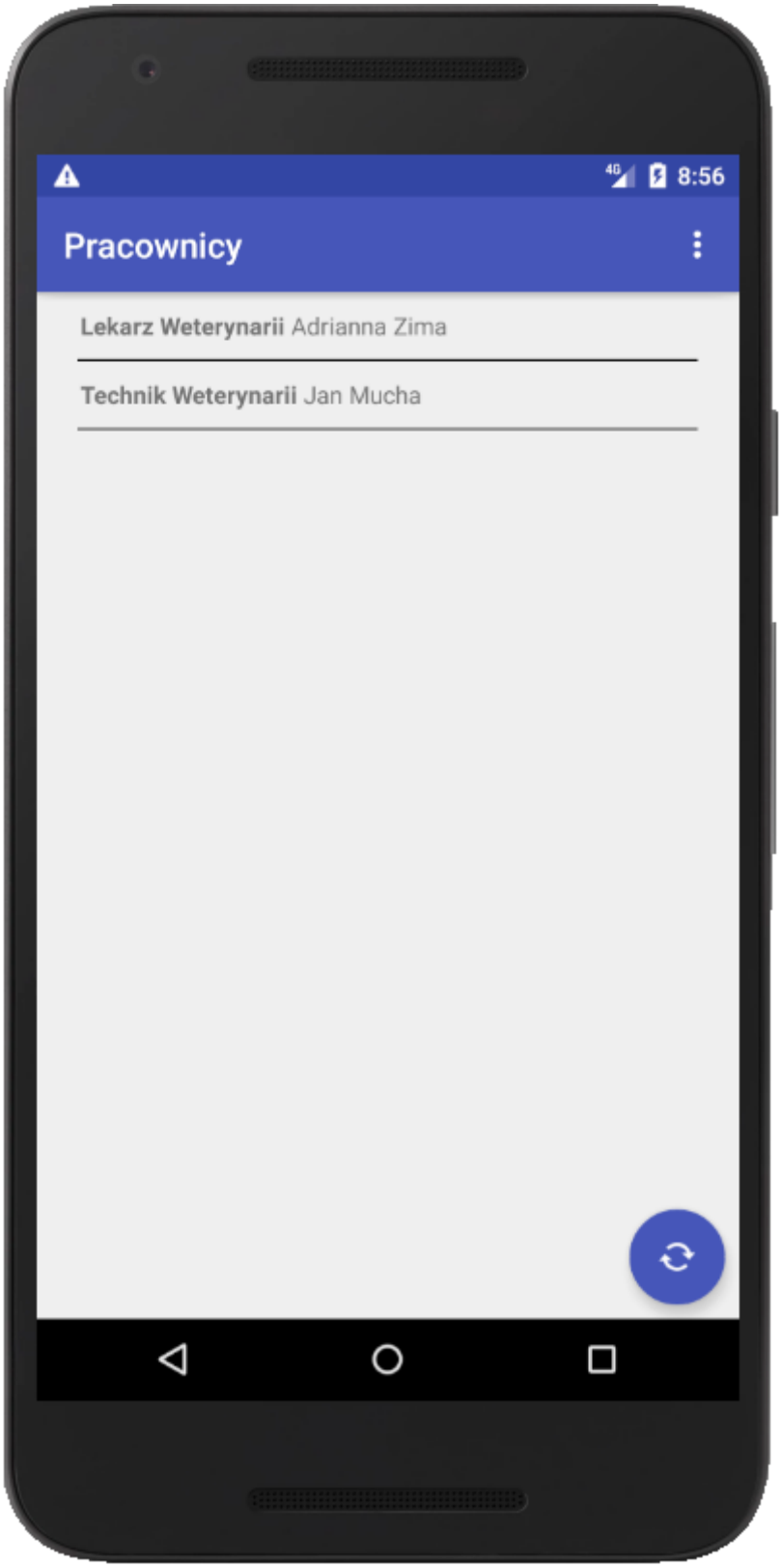
Przykładowe widoki zdziałającej aplikacji mobilnej wykorzystującej testowe dane:



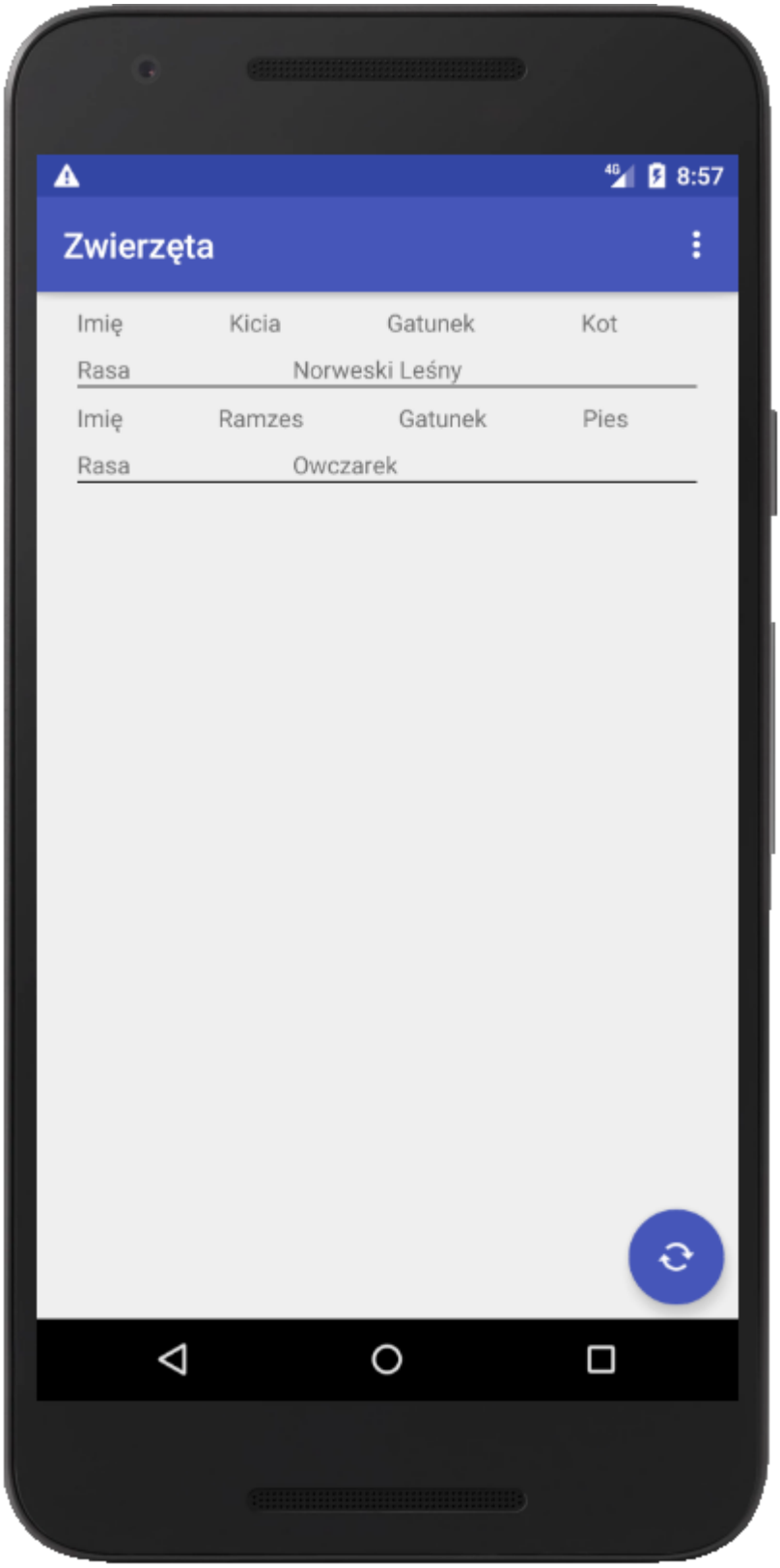
Ilustracja 7: Logowanie



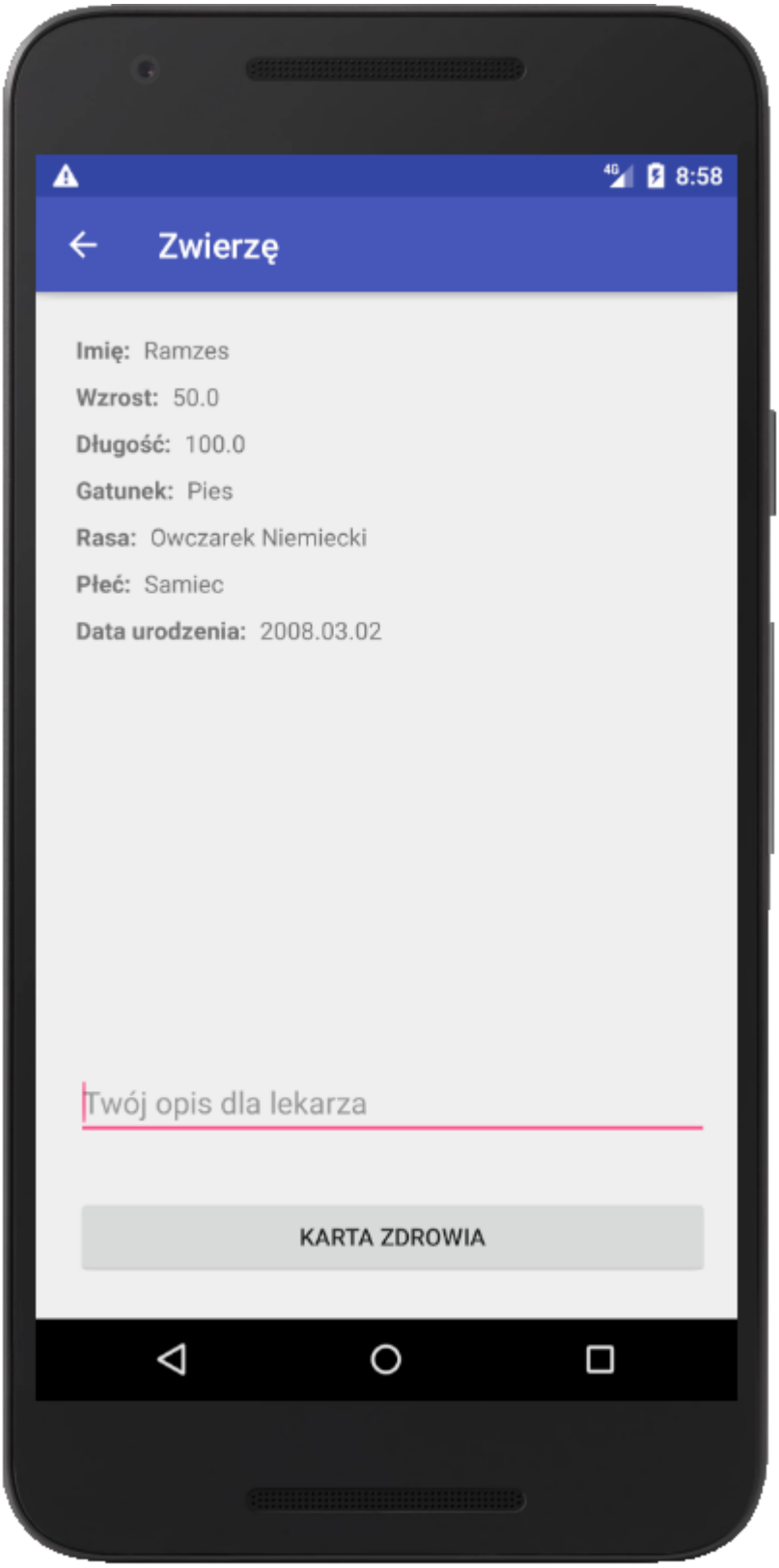
Ilustracja 8: Widok kliniki



Ilustracja 9: Lista pracowników

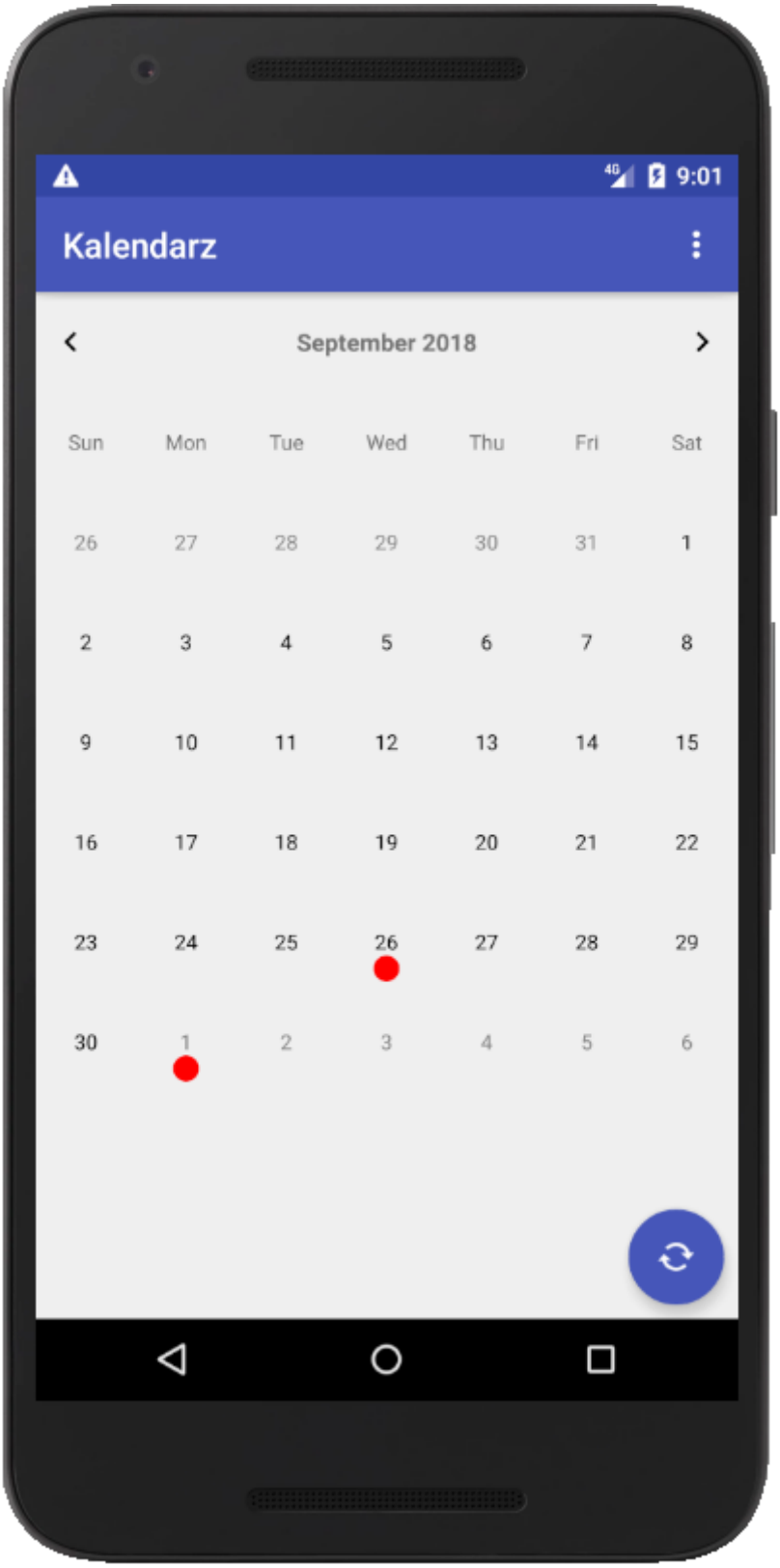


Ilustracja 10: Zwierzęta

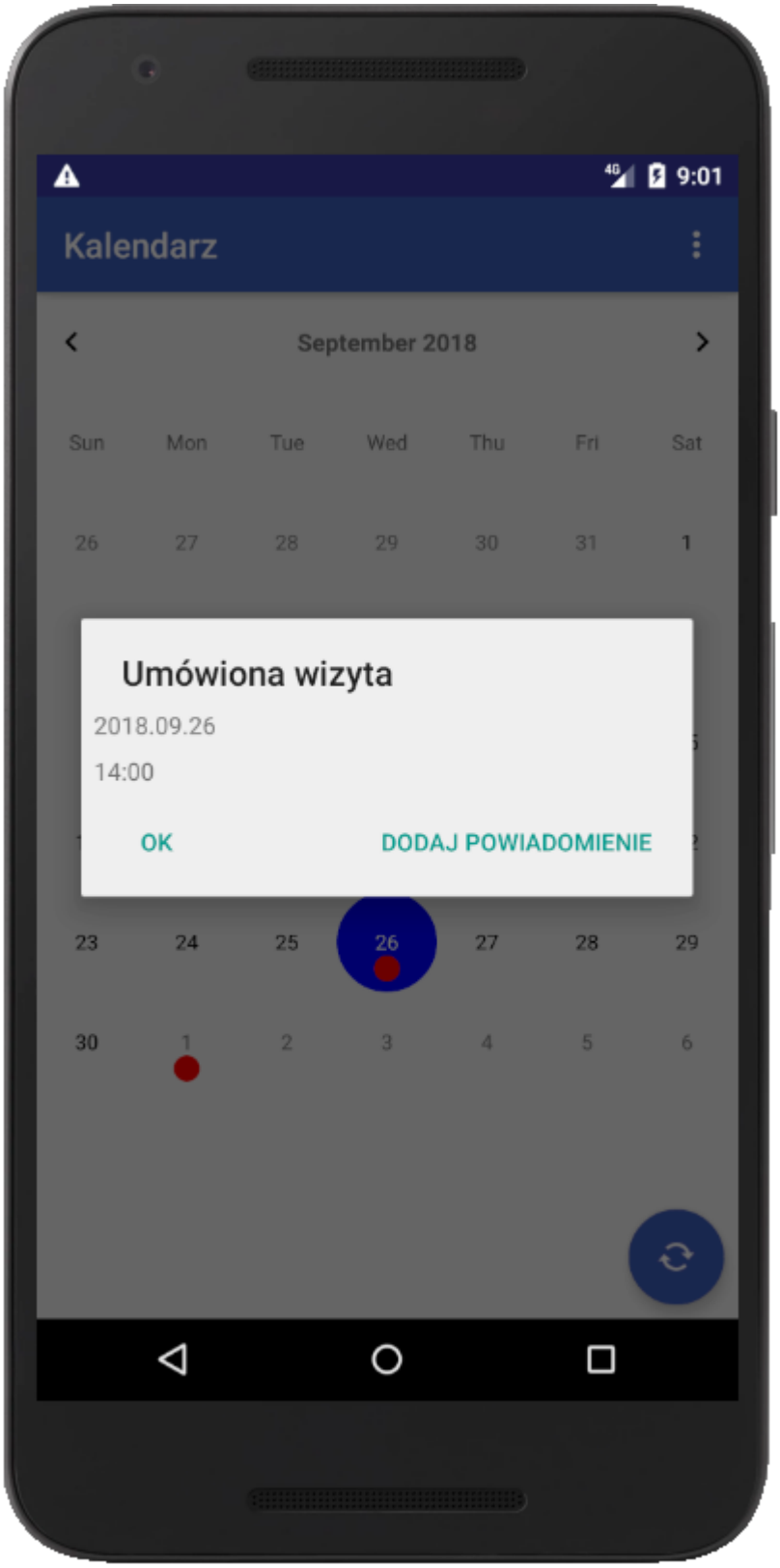


Ilustracja 11: Szczegóły zwierzęcia

Ilustracja 12: Karta zdrowia



Ilustracja 13: Widok kalendarza



Ilustracja 14: Umówiona wizyta

10. Spis ilustracji

Spis Ilustracji

11. Spis diagramów

Spis Diagramów

12. Spis kodu

Spis Diagramów