POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Inżynierii Oprogramowania

Inżynieria Oprogramowania i Baz Danych

Magdalena Popek	Nr albumu 12988
Paulina Gruba	Nr albumu 13078
Piotr Prystupa	Nr albumu 12463
Jacek Muszyński	Nr albumu 12662
Dominik Deja	Nr albumu 12898
Paulina Stasiów	Nr albumu 12955
Patryk Pohnke	Nr albumu 12999
Szymon Ritz	Nr albumu 12910
Paweł Walesiak	Nr albumu 12828

System Informatyczny Do Obsługi Stajni - SIDOS

Praca inżynierska

dr inż. Mariusz Trzaska

Warszawa, luty, 2018

Streszczenie

Celem ninejszej pracy inżnierskiej było stworzenie systemu informatycznego wspomagającego zarządzanie ośrodkiem jeździeckim. System Informatyczny Do Obsługi Stajni (SIDOS) składa się z aplikacji webowej, napisanej w języku C# wykorzystując framework Microsoft ASP.NET MVC oraz aplikacji mobilnej napisanej w języku Java na platformę Android. Założeniem systemu było wsparcie procesów biznesowych związanych z zarządzaniem ośrodkiem jeździeckim.

Słowa kluczowe

.net, android studio, C#, asp.net, mvc, jquery, ajax, html, javascript, css, entity framework, aplikacja webowa, aplikacja mobilna, koń, ośrodek jeździecki, stadnina, system do zarządzania, zarządzanie pracownikami stadniny, system do zarządzania ośrodkiem, zadania, stajnia online

Spis treści

1 Wstęp

Istnieje wiele aplikacji do zarządzania stadniną, jednak żadna z nich nie jest dostępna w języku polskim. W związku z tym część polskich stadnin przechowuje dane dotyczące klientów, koni, zajęć oraz zadań w wersji papierowej, w Excelu lub kalendarzu Google.

Rozwiązania te mają swoje wady, między innymi:

- Dane zapisane na papierze lub w Excelu dostępne w jednym miejscu i dla ograniczonej liczby odbiorców w tym samym czasie;
- Informacje tak przechowywane są nietrwałe (medium łatwo może ulec zniszczeniu, a dane przechowywane w ten sposób zazwyczaj nie są zapisywane w żadnym innym miejscu);
- Kalendarz udostępniany przez Google umożliwia zapisywanie zajęć i spotkań, jednak nie jest przystosowany do przechowywania większej ilości informacji;
- Wymienione media pozwalają na przechowywanie niespójnych danych.

W niniejszej pracy zostały rozwiązane m.in. wymienione problemy poprzez stworzenie przystępnego systemu w języku polskim.

1.1 Ośrodek jeździecki

Ośrodek jeździecki jest miejscem, w którym można rozpocząć naukę jazdy konnej lub doskonalić swoje umiejętności zarówno samodzielnie, na własnym koniu, bądź pod okiem instruktora lub trenera jeździectwa. Istnieje również możliwość zakwaterowania swojego konia w jednym z dostępnych boksów czy wydzierżawienia[1] konia na pewien okres.

W ośrodku istnieje potrzeba przechowywania danych koni, ich właścicieli, klientów i pracowników. Każdy trener i instruktor ma swój grafik prowadzonych jazd, a konie nie powinny uczestniczyć w więcej niż określonej liczbie jazd w ciągu dnia. Należy również wiedzieć, który koń jest prywatny, który należy do szkółki jeździeckiej, a który aktualnie jest w dzierżawie.

[1] Wydzierżawiający (właściciel konia, np. ośrodek jeździecki) zobowiązuje się oddać dzierżawcy rzecz (w tym przypadku konia) do używania i pobierania pożytków przez czas oznaczony, a dzierżawca zobowiązuje się płacić wydzierżawiającemu określony w umowie czynsz. W praktyce oznacza to, że dzierżawca staje się niejako właścicielem konia na czas trwania umowy. [1]

Ważne jest, żeby przy większej liczbie koni jasne było, jak dbać o każdego z nich – niektóre mogą potrzebować leków lub specjalnego pożywienia. U części z nich należy stosować się do zaleceń weterynarza, a inne mogą mieć zaplanowane wizytę lub zabieg.

Wszystkie te wymagania generują problemy organizacyjne, które trzeba rozwiązać w najbardziej optymalny sposób.

1.2 Cel pracy

Celem niniejszej pracy inżynierskiej było stworzenie systemu informatycznego do obsługi ośrodków jeździeckich, składającego się z aplikacji webowej oraz mobilnej. System ma na celu wspierać procesy związanie z zarządzaniem stadniną oraz komunikację pomiędzy kierownikiem a pracownikami i ułatwić wypełnianie obowiązków.

Projekt został zrealizowany w ramach specjalizacji Inżynieria Oprogramowania i Baz Danych w Polsko-Japońskiej Akademii Technik Komputerowych.

1.3 Rezultat pracy

Rezultatem pracy inżynierskiej jest prototyp systemu do zarządzania stadniną oraz niniejsze opracowanie.

System wspomaga działanie organizacji jaką jest stajnia. Realizuje to m.in. poprzez funkcjonalności, które wspomagają pracowników stajni np. kierownik może przypisywać zadania poszczególnym pracownikom, edytować je i usuwać. Instruktor może kontrolować lekcje, a weterynarz sprawdzać swoje wizyty, które musi wykonać albo kiedyś wykonał.

System umożliwia osobom niezalogowanym przeglądanie najważniejszych informacji o ośrodku na stronie głównej. Do skorzystania ze wszystkich funkcjonalności dostępnych dla klientów wymagane jest założenie konta i zalogowanie się. Po zalogowaniu osoba odwiedzająca stronę staje się użytkownikiem i otrzymuje dostęp do kokpitu. W tej części strony może dokonać m.in. zapisu na daną jazdę, przejrzeć kalendarz zaplanowanych jazd i więcej aktualności związanych ze stajnią.

Jeśli użytkownik posiada telefon komórkowy z systemem Android, może też skorzystać z wersji mobilnej systemu.

Elementy strony dostępne dla użytkowników przed zalogowaniem

1. Ogólne informacje o stajni
2. Aktualności
3. Cennik
4. Lokalizacja stajni
5. Informacje o koniach i pracownikach

Elementy strony dostępne po zalogowaniu:

1. Lekcje – tworzenie, edytowanie
2. Jazdy – tworzenie, edytowanie, zapisywanie się i odwoływanie
3. Zadania – tworzenie, edytowanie i przypisywanie zadań poszczególnym pracownikom
4. Konie – tworzenie i edytowanie
5. Wizyty weterynaryjne – tworzenie, edytowanie i przypisywanie wizyt poszczególnym weterynarzom
6. Dzierżawy koni – tworzenie i edytowanie
7. Pracownicy – tworzenie i edytowanie informacji o nich
8. Kalendarz z poszczególnymi zaplanowanymi lekcjami
9. Aktualności dostępne dla pracowników

Elementy w aplikacji mobilnej przed zalogowaniem:

1. Przeglądanie profili instruktorów
2. Przeglądanie koni
3. Wyszukiwanie koni na podstawie cech

Elementy aplikacji mobilnej po zalogowaniu:

1. Przeglądanie, akceptowanie i odrzucanie zadań
2. Przeglądanie lekcji, na które jest zapisana osoba
3. Wysłanie zgłoszenia do weterynarza
4. Przyjmowanie, odrzucanie i zmiana zgłoszeń o problemie zdrowotnym

1.4 Organizacja pracy

W rozdziale drugim autorzy opisują stan sztuki.

Rozdział trzeci zawiera propozycję systemu: wymagania funkcjonalne i niefunkcjonalne systemu.

W rozdziale czwartym opisane są narzędzia i technologie użyte w pracy.

Rozdział piąty poświęcony jest przedstawieniu aplikacji – założenia architektoniczne, interfejsy użytkowników, opis komunikacji z poziomu aplikacji oraz sposób wsparcia pracowników stadniny.

Rozdział szósty zawiera opis fazy testów.

Rozdział siódmy stanowi podsumowanie pracy. Przedstawione są w nim zrealizowane założenia, napotkane trudności oraz możliwe kierunki rozwoju.

W załączniku A przedstawiono organizację pracy grupy.

W załączniku B opisano zmiany wprowadzone w trakcie fazy projektowania.

2 Istniejące systemy do zarządzania stadniną

Na rynku istnieje wiele systemów, głównie w języku angielskim, służących do wspomagania zarządzania ośrodkiem jeździeckim. W Polsce nie są one tak popularne, gdyż w większości przypadków stadniny korzystają z dedykowanych systemów stworzonych specjalnie na potrzeby danego ośrodka. Żadna ze stajni, z którą się kontaktowaliśmy, nie chciała udzielić nam informacji dotyczących wykorzystywanego oprogramowania, w związku z czym na potrzeby tej pracy omówione zostaną trzy ogólnodostępne systemy stworzone poza Polską:

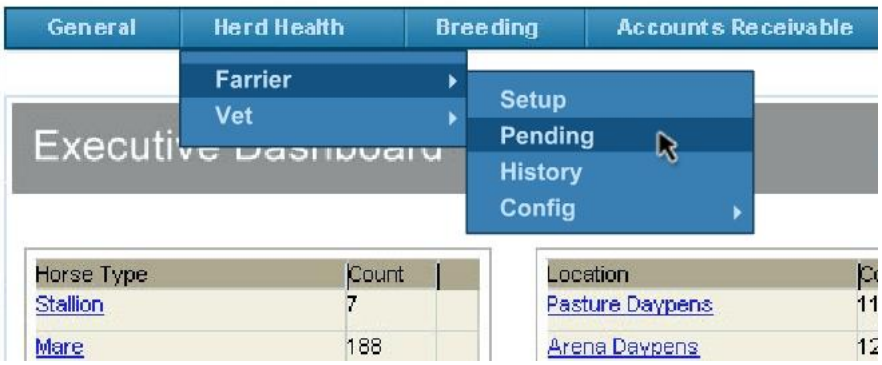
1. PaddockPro
2. HorseCount
3. BarnManager

2.1 PaddockPro

PaddockPro jest systemem stworzonym w 2006 roku przez amerykańską firmę Alua Software [2]. Zgodnie z opisem system ma obsługiwać wszystkie potrzeby biznesowe dotyczące koni, takie jak weterynarz, kowal, hodowla, przygotowywanie sprzedaży oraz wystawianie rachunków. System zapewnia:

1. Obsługę rachunków i faktur
2. Zarządzanie hodowlą
3. Bazę danych klientów
4. Bazę danych koni
5. Śledzenie wydatków
6. Zarządzanie karmieniem
7. Zestawienia przychodów
8. Dokumentację medyczną koni

PaddockPro jest kompleksowym systemem umożliwiającym przechowywanie wszystkich informacji o koniach oraz ich właścicielach, wizytach weterynaryjnych oraz pracach kowala. Aby umówić wizytę weterynarza czy kowala wystarczy wejść w odpowiednią zakładkę w menu, przedstawioną na rysunku 1, a następnie wybrać jedną spośród interesujących nas opcji. [3] Rysunek 2 przedstawia podgląd umówionych wizyt weterynaryjnych.



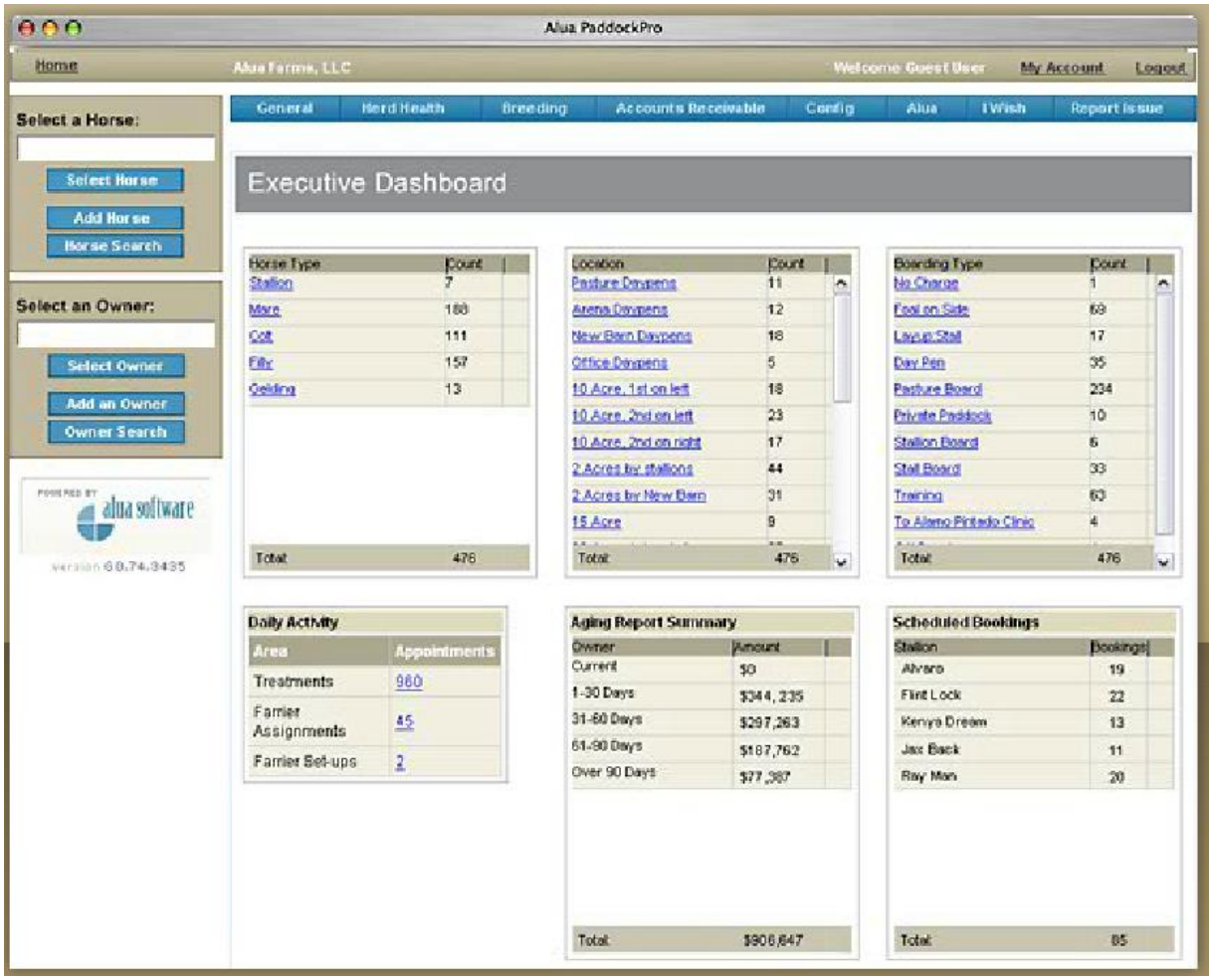
Rysunek 1. Część menu zawierająca opcje wyboru elementów związanych ze zdrowiem koni (PaddockPro)

Rysunek 2. Tabela umówionych wizyt weterynaryjnych (PaddockPro).

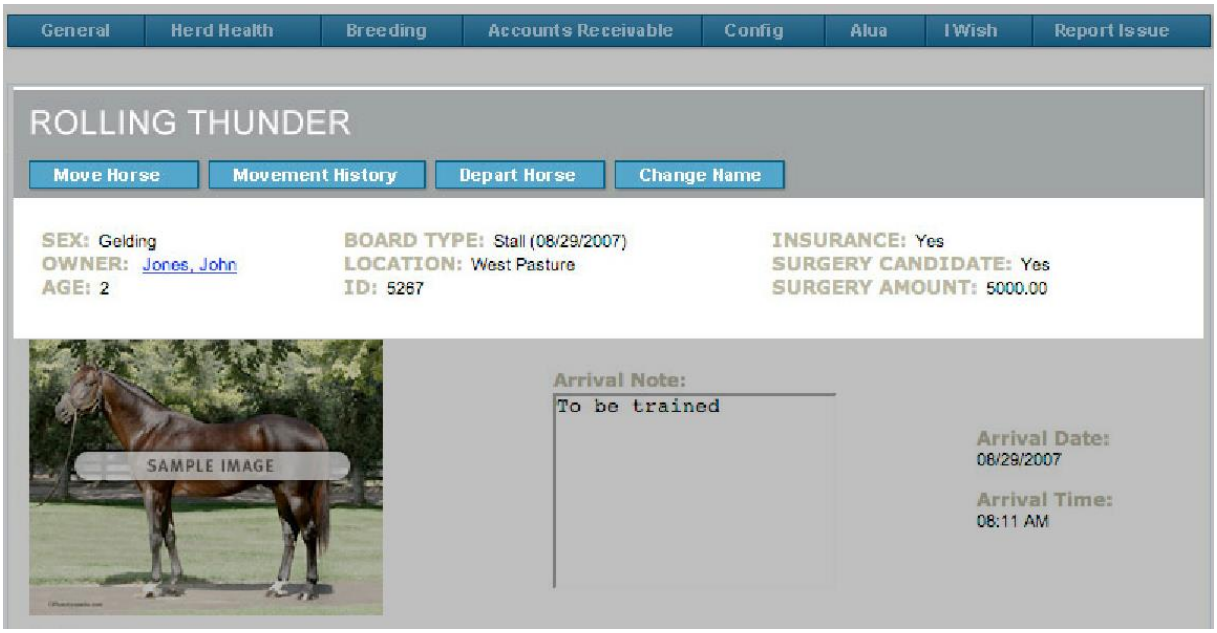
Dodatkowym atutem PaddockPro jest możliwość monitorowania finansów stadniny. Tak samo, jak w przypadku kontrolowania stanu wizyt weterynarza i kowala, wystarczy wejść w odpowiednią zakładkę menu. Rysunek 3 przedstawia menu płatności.

Rysunek 3. Menu płatności (PaddockPro).

Niewątpliwą zaletą systemu PaddockPro jest łatwa dostępność najważniejszych elementów. Strona główna zapewnia szybki wgląd do kluczowych informacji o stajni, m.in. liczbie koni danej płci, umówionych spotkaniach czy stanie poszczególnych lokalizacji. Interfejs umożliwia sprawne wyszukanie konia lub właściciela oraz podgląd ich szczegółowego profilu. Rysunek 4 przedstawia kokpit systemu, natomiast Rysunek 5 podgląd profilu konia. [4]



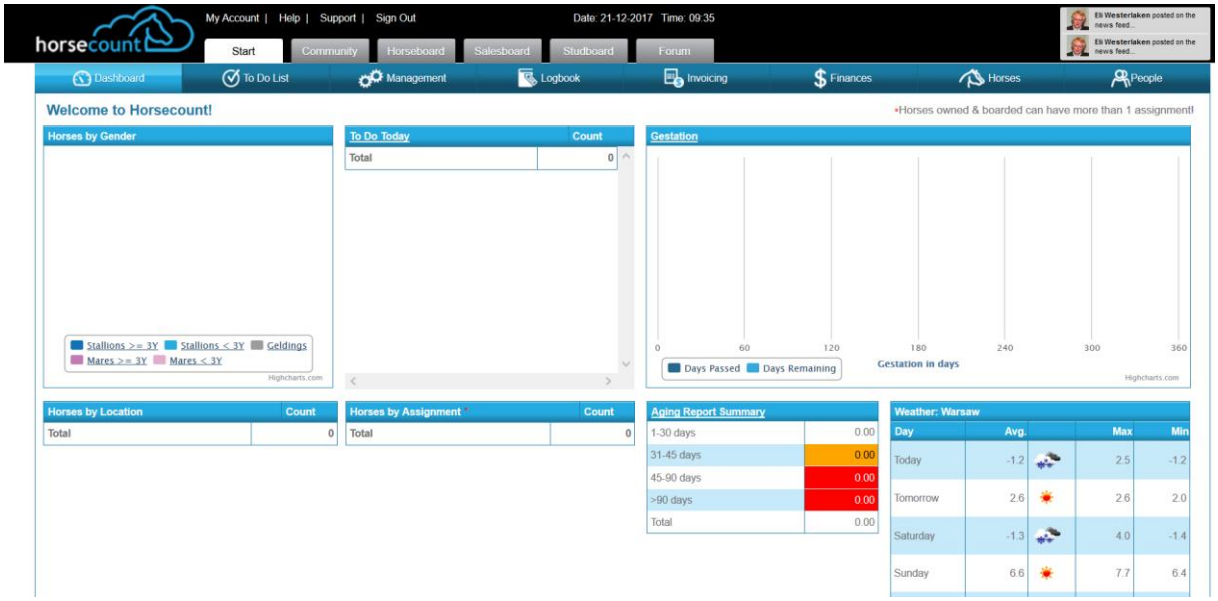
Rysunek 4. Kokpit systemu (Dashboard) (PaddockPro)



Rysunek 5. Profil konia (PaddockPro)

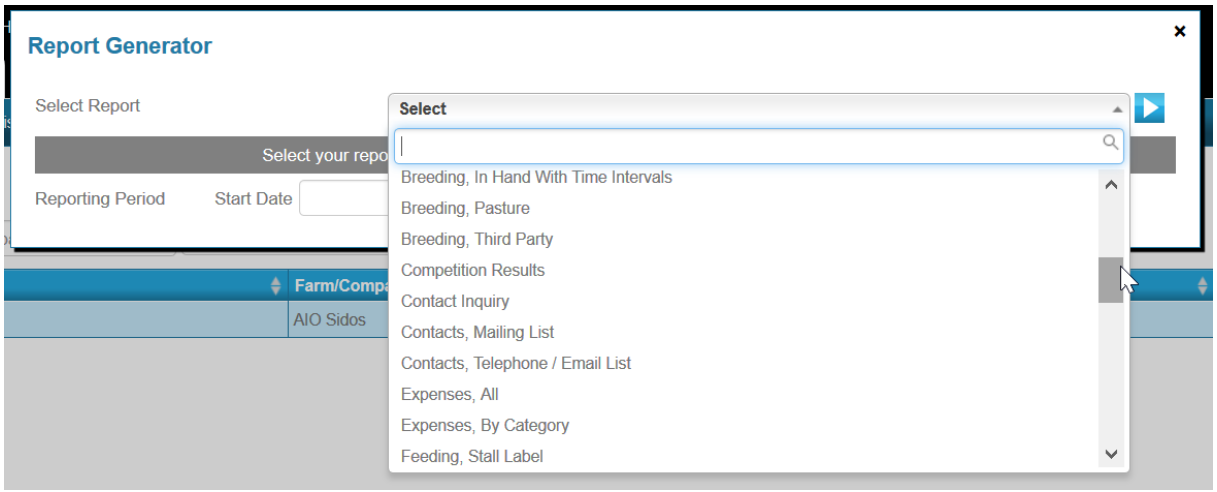
2.2 HorseCount

HorseCount został stworzony w 2012 roku przez holenderską firmę animalcount BV. Jest aplikacją (webową i mobilną) dla właścicieli koni oraz stadnin, pozwalającą na zarządzanie swoimi zwierzętami, codziennymi czynnościami związanymi z nimi, jak i finansami. Dodatkową zaletą HorseCount jest społeczność, z którym możemy dzielić się naszymi zdjęciami, osiągnięciami, wymieniać wiadomości, a także zakładać grupy czy tworzyć i wyszukiwać wydarzenia. Dodatkowymi funkcjonalnościami HorseCount jest przykładowo tablica ogłoszeń sprzedaży, gdzie każdy zarejestrowany użytkownik może wystawić swojego konia na sprzedaż. Wszystkie ogłoszenia są widoczne dla pozostałych użytkowników HorseCount [5]. Rysunek 6 przedstawia stronę główną systemu.



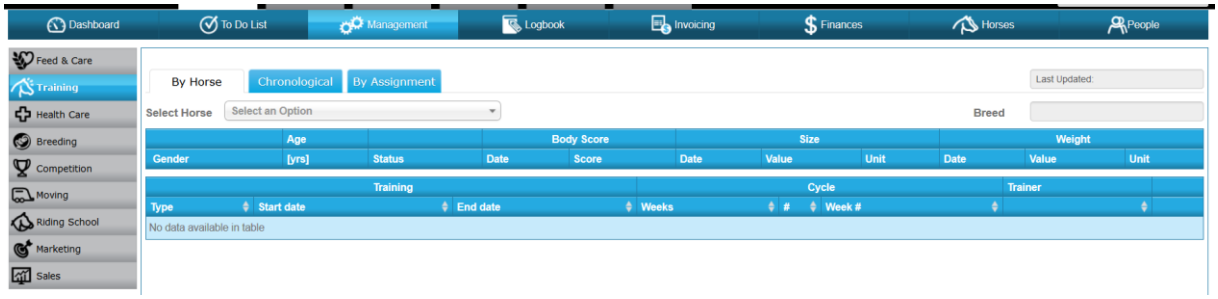
Rysunek 6. Strona główna (HorseCount)

System umożliwia również tworzenie raportów z elementów, których zestawieniem możemy być zainteresowani, np. kontakty, wydatki. Rysunek 7 przedstawia kilka z możliwych raportów do wyboru.



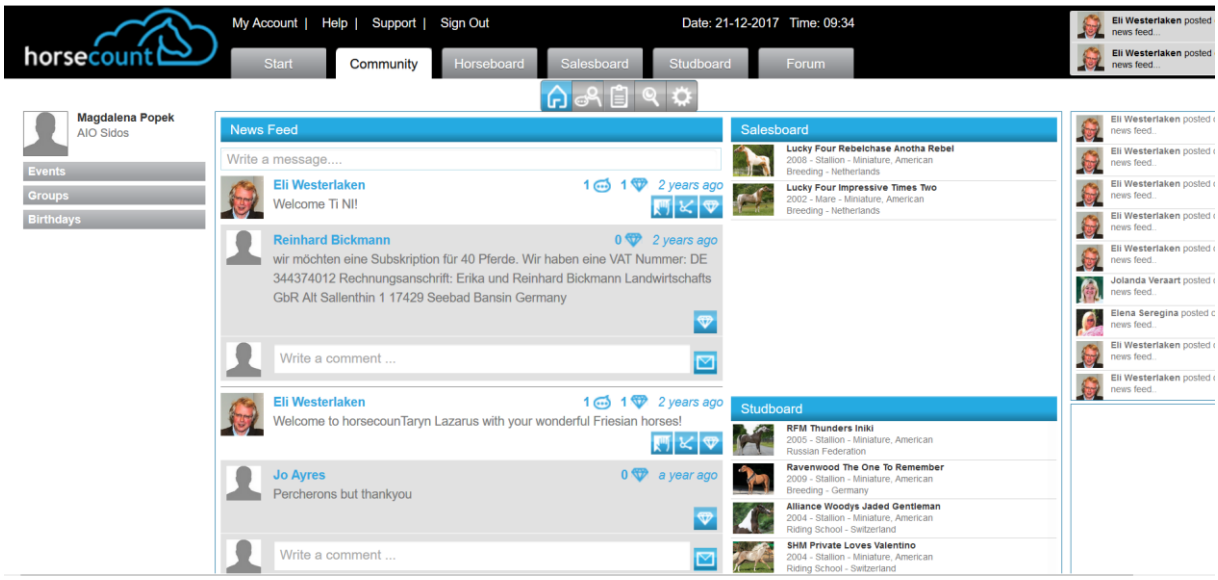
Rysunek 7. Przykładowe raporty (HorseCount)

Dzięki HorseCount możemy dokładnie kontrolować aktualny stan treningów naszych koni czy ich zdrowie. Dodatkowo mamy wgląd w zaplanowane zawody, zajęcia w szkółce jeździeckiej, sprzedaże czy przeprowadzki koni. Rysunek 8 przedstawia stronę koni w treningu.



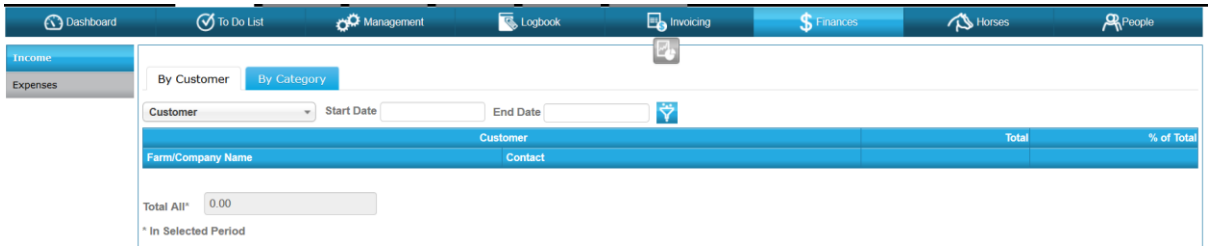
Rysunek 8. Lista koni w treningu (HorseCount)

Dużą zaletą HorseCount jest społeczność umożliwiająca kontakt z innymi użytkownikami systemu, zadawanie pytań na forum czy dzielenie się zdjęciami. Rysunek 9 prezentuje zakładkę społeczności.

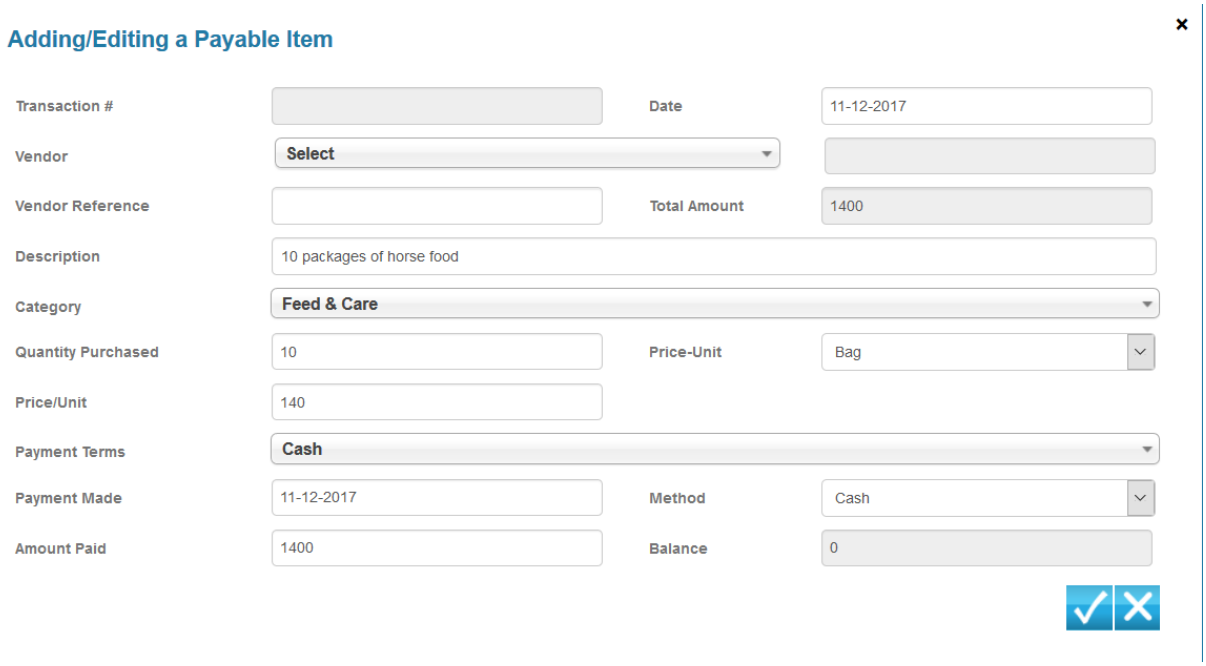


Rysunek 9. Zakładka społeczności (HorseCount)

HorseCount pozwala na zarządzanie wydatkami i przychodami poprzez dedykowany panel finansowy. Rysunek 10 przedstawia stronę zarządzania płatnościami. Rysunek 11 przedstawia formularz dodania płatności. [6]



Rysunek 10. Strona zarządzania płatnościami (HorseCount)

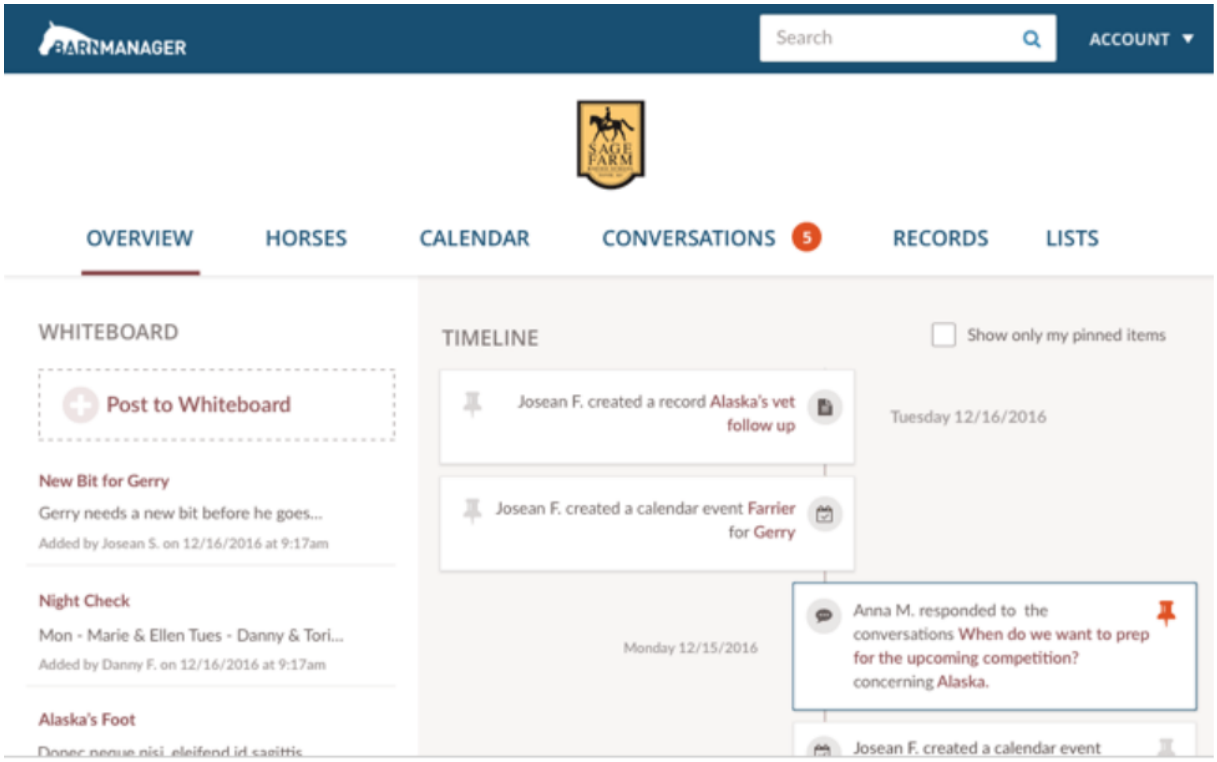


Rysunek 11. Formularz dodania płatności (HorseCount)

2.3 BarnManager

BarnManager jest rozwiązaniem opartym na Chmurze [7], dostarczającym narzędzia dla właścicieli koni i ośrodków jeździeckich ułatwiające zarządzanie codziennymi obowiązkami. Program ma za zadanie ułatwić przechowywanie danych dotyczących wielu aspektów opieki nad koniem, jak i ułatwić zarządzanie małym biznesem. Dostęp do aplikacji można uzyskać zarówno przy pomocy komputera, jak i tabletu czy smartphone. BarnManager zapewnia śledzenie zapisów o opiece nad koniem oraz umożliwia zarządzanie klientami i pracownikami. Przechowywanie informacji o stanie zdrowia pozwala m.in. na przypominanie klientom, kiedy ostatni raz ich koń był szczepiony. Poprzez system możemy m.in. planować wizyty weterynarza oraz kowala, zapisywać się na jazdy, prowadzić konwersacje z klientami i zespołem, a także ustawiać powiadomienia w celu przypomnienia o wykonaniu zadania. [8]

W przeciwieństwie do PaddockPro i HorseCount, BarnManager nie zapewnia śledzenia przychodów i wydatków. Twórcy systemu zaznaczyli, że dokonywanie i otrzymywanie płatności, raportowanie finansów będą możliwe w przyszłości. Chwilowo na stronie internetowej BarnManager wszelkie opcje związane z zarządzaniem biznesem są oznaczone jako nadchodzące. [9] Rysunek 12 przedstawia stronę główną systemu BarnManager.



Rysunek 12. Strona główna (BarnManager)

2.4 Podsumowanie

Wszystkie omówione rozwiązania mają na celu ułatwienie prowadzenia ośrodka jeździeckiego. Częścią wspólną PaddockPro, HorseCount, BarnManager oraz naszego SIDOS jest baza danych koni, klientów i pracowników, możliwość śledzenia stanu zdrowia konia, umawiania wizyt weterynarza i kowala, zapisywania się na zajęcia jazdy konnej. Jednak opisane rozwiązanie są systemami do obsługi biznesu, dostępu do których nie ma nikt poza zarejestrowanymi użytkownikami. SIDOS jako aplikacja webowa odpowiada zarówno na potrzebę posiadania reprezentatywnej strony internetowej dla odwiedzających, jak i potrzebę minimalizacji prawdopodobieństwa błędu podczas komunikacji pracowników między sobą czy pracowników z klientami. SIDOS zmniejsza do minimum potrzebę komunikacji poza systemem, m.in. dzięki możliwości złożenia wniosku o dzierżawę konia czy opcji zlecania zadań, dzięki czemu eliminujemy możliwość przeoczenia czy złego zapamiętania informacji przez pracownika. Pomimo braku pewnych złożonych funkcjonalności finansowych czy hodowlanych, SIDOS, w przeciwieństwie do wymienionych w tym rozdziale systemów, pomaga w wykluczeniu błędu ludzkiego w przekazywaniu informacji między pracownikami i klientami oraz przedstawieniu stajni poprzez umieszczoną w sieci stronę internetową.

3 Propozycja systemu

W rozdziale tym opisujane zostało rozwiązanie do wdrożenia Systemu Informatycznego Do Obsługi Stajni, wymagania funkcjonalne i niefunkcjonalne. Uzupełnieniem rozdziału są diagramy przypadków użycia i diagram klas. W niniejszym rozdziale nie zostały opisane źródła danych poszczególnych pól i zasad walidacji. Opis taki powstanie w Fazie Projektowania. SIDOS ma być kompletnym narzędziem do obsługi stajni - zarówno dla właściciela, pracowników i współpracowników, jak i klientów.

3.1 Założenia

SIDOS ma być kompletnym narzędziem do zarządzania stajniami. W swoim założeniu system ma być elastyczny, dzięki czemu idealnie dopasuje się do zróżnicowanych wymagań i potrzeb różnych stajni. Jego celem jest automatyzacja wielu czynności (m.in. umawianie zajęć jazdy konnej), co spowoduje usprawnienie komunikacji na poziomie pracowników i klientów oraz pozwoli na przechowywanie wszystkich danych dotyczących funkcjonowania stadniny w jednym miejscu.

Jedną z podstawowych funkcjonalności systemu jest możliwość umawiania zajęć jazdy konnej zarówno z profilu klienta (klient z wykorzystaniem systemu umawia się na zajęcia) oraz z profilu pracownika (gdy osoba zainteresowana ofertą stajni nie posiada konta w systemie pracownik ma możliwość zapisania takiej osoby na zajęcia). W systemie przechowywane będą informacje o zajęciach zaplanowanych, odbytych oraz odwołanych (razem z przyczyną odwołania).

SIDOS umożliwi klientom, pracownikom oraz właścicielom koni złożenie zgłoszenia do weterynarza za pomocą specjalnego formularza zawartego w systemie. Będzie to możliwe również za pomocą aplikacji mobilnej. Podstawowym elementem zgłoszenia będzie zaznaczenie na szkicu miejsca na ciele konia, którego dotyczy zgłoszenie oraz dodanie opisu. Opcjonalnie do zgłoszenia można dołączyć zdjęcie w celu ułatwienia dokonania wstępnej diagnozy. System umożliwi również umawianie regularnych wizyt. Dzięki książeczce zdrowia on-line wszystkie osoby uprawnione do wglądu będą miały możliwość w każdej chwili zapoznać się z historią zdrowia konia.

System ma na celu ułatwienie pracy pracowników, dzięki możliwości samodzielnego zarządzania swoimi profilami oraz wprowadzania zmian w grafiku. Komunikacja między

pracownikami i klientami odbywać się będzie poprzez wiadomości wysyłane w obrębie systemu, dzięki czemu całość będzie przechowywana w jednym miejscu i komunikacja może odbywać się jednym kanałem, co spowoduje jej usprawnienie.

Weterynarze korzystający z systemu będą w prosty sposób otrzymywać zgłoszenia dotyczące koni oraz będą mieć ułatwiony kontakt ze zgłaszającym.

Przewidziana jest aplikacja mobilna, która będzie posiadała najważniejsze funkcjonalności aplikacji webowej (takie jak możliwość wysyłania zgłoszeń do weterynarza, umawiania na zajęcia czy wysyłania i odbierania wiadomości).

3.2 Definicje

Aplikacja mobilna – dedykowana aplikacja na urządzenia mobilne (Android), dająca dostęp do wybranych funkcji systemu i korzystająca z części funkcjonalności serwerowej.

Instruktor – osoba posiadająca uprawnienia instruktorskie.

Kierownik – osoba zarządzająca stajnią.

Klient – osoba chcąca korzystać z oferty stajni posiadająca konto w systemie.

Osoba dzierżawiąca – osoba dzierżawiąca konia od stajni lub osoby fizycznej, może nią być Klient lub Pracownik.

Podsystem czasu – system do zarządzania cyklicznymi zmianami w systemie.

Pracownik – pracownik stajni, nie musi być zatrudniony przez stajnię, może z nią współpracować.

Stajenny – osoba zajmująca się końmi, zatrudniona przez stajnie.

Użytkownik – osoba zarejestrowana w systemie.

Weterynarz – osoba, która otrzymała prawo do wykonywania zawodu weterynarza, posiadająca kwalifikacje do badania i leczenia zwierząt.

Właściciel konia – osoba trzymająca konia w stajni, może to być Klient lub Pracownik.

3.3 Użytkownicy systemu

3.3.1 Klient

1. Przechowywane dane:
    a. Imię
    b. Nazwisko
    c. Data urodzenia
    d. Adres e-mail
    e. Numer telefonu
    f. Zdjęcie profilowe
    g. Hasło
    h. Data ostatniej zmiany hasła
2. Dostępne funkcjonalności:
    a. Edycja profilu
    b. Wysyłanie i odbieranie wiadomości
    c. Umawianie i odwoływanie zajęć
    d. Przeglądanie zajęć, na które jest zapisany

Ad.3.3.1.2.a - Klient ma możliwość edycji profilu poprzez wejście w zakładkę „Mój profil" oraz wybranie odpowiedniej opcji na stronie. Klient może zmienić swoje hasło (w dowolnym wybranym przez siebie momencie, minimum pół roku po ostatniej zmianie hasła. Jeśli Klient sam nie zmieni hasła w odpowiednim czasie pojawi się stosowne powiadomienie), zdjęcie profilowe oraz dane kontaktowe – numer telefonu oraz adres e-mail. System nie zakłada możliwości zmiany informacji takich jak dane osobowe (imię i nazwisko, data urodzenia).

Ad.3.3.1.2.b - Klient ma możliwość zarządzania swoimi wiadomościami. Z poziomu skrzynki nadawczej i odbiorczej klient ma możliwość usuwania wiadomości. Poprzez SIDOS Klient ma możliwość wysyłania wiadomości do wszystkich zarejestrowanych użytkowników systemu.

Ad.3.3.1.2.c - Aby umówić się na zajęcia Klient musi wejść w zakładkę „Umów się na zajęcia" dostępną w jego panelu. Następnie w wyświetlonym widoku wybiera zajęcia, na które chce się umówić. Po kliknięciu przycisku „umów się na zajęcia" system wyświetli okno z możliwością wpisania komentarza do jazdy.

Ad.3.3.1.2.d - Na profilu Klienta dostępny będzie kalendarz, gdzie będzie miał podgląd na zajęcia, na które jest zapisany. Gdy Klient wejdzie w konkretne zajęcia pokażą mu się ich szczegóły (godzina, osoba, z którą zajęcia się odbędą, koń (jeśli jest przypisany)). W szczegółach zajęć dostępna będzie opcja „Odwołaj zajęcia". Gdy Klient ją wybierze system wyświetli okno, w którym będzie mógł wprowadzić opis lub powód rezygnacji i po zaakceptowaniu, powiadomienie to zostanie wysłane do pracownika, z którym miały odbyć się te zajęcia.

3.3.2 Właściciel konia

1. Dostępne funkcjonalności jak u Klienta plus dodatkowe:

    a. Zakładka „Moje konie"
    b. Edytowanie profilu swojego konia
    c. Umawianie się na zajęcia ze swoim koniem
    d. Przeglądanie historii wizyt swojego konia lub konia, którego ma w dzierżawie

Ad.3.3.2.1.a - W zakładce „Moje konie" przechowywana jest lista koni właściciela (we wszystkich stadninach korzystających z systemu). Z listy tej Właściciel może wejść na profil swojego konia i sprawdzić archiwum, w którym znajduje się lista koni, które przez pewien okres były oznaczone jako jego własność oraz informacje, dlaczego dany koń przestał być własnością („zdechł", „sprzedany"). Właściciel konia ma wgląd w książeczkę zdrowia swojego zwierzęcia on-line oraz może za pomocą systemu umawiać wizytę u weterynarza, która nie jest nagłym zgłoszeniem.

Ad.3.3.2.1.b - Właściciel konia ma możliwość ukrycia profilu swojego konia tak, że aktorzy Gość i Klient nie będą mieli podglądu jego profilu. Opcję taką może w każdym momencie wyłączyć, upubliczniając profil zwierzęcia.

Ad.3.3.2.1.c - Gdy Właściciel konie będzie chciał się umówić na zajęcia, proces będzie wyglądał tak, jak w przypadku aktora Klient. Właściciel jednak będzie miał możliwość zaznaczenia chęci uczestnictwa w zajęciach na swoim koniu. Jeżeli posiada więcej niż jednego konia, będzie mógł wybrać go z listy.

Ad.3.3.2.1.d – Właściciel konia ma wgląd w historię wizyt swojego lub dzierżawionego konia u weterynarza. Jeśli Klient dzierżawcą, razem z końcem dzierżawy kończy się dostęp do tych informacji.

3.3.3 Osoba dzierżawiąca

1. Przechowywane dane jak u Klienta plus dodatkowe:

    a. Cena za miesiąc
    b. Data od kiedy dzierżawi
    c. Data do kiedy dzierżawi
    d. Uwagi

3.3.4 Pracownik

1. Przechowywane dane takie jak u Klienta plus dodatkowe:

    a. Posiadane certyfikaty
    b. Od kiedy zatrudniony
    c. Do kiedy zatrudniony
    d. Galeria zdjęć

2. Dostępne funkcjonalności jak u Klienta plus dodatkowe:

    a. Edytowanie swojej galerii
    b. Dodawanie i usuwanie posiadanych certyfikatów
    c. Edytowanie swojego grafiku
    d. Zmiana statusu zadania / aktualizacja zadania
    e. Przeglądanie informacji o koniu
    f. Zmiana danych kontaktowych

Ad.3.3.4.1– Wszystkie te informacje będą dostępne na profilu Pracownika; widoczne dla osób odwiedzających stronę.

Ad.3.3.4.2.d - Pracownik może zmienić status swojego zadania jako wykonane, zaakceptowane lub całkowicie je odrzucić.

Ad.3.3.4.2.e - Edycja grafiku poprzez umawianie zajęć (wymagane dane: uczestnicy, konie, data, godzina). Pracownik może także edytować swój grafik, poprzez zmianę statusów zajęć („przeniesione", „odrzucone"), może dodać opis spotkania, może je przenieść na inną godzinę lub inny dzień. Pracownik w swoim grafiku ma możliwość zmiany uczestników zapisanych w grafiku – może ich zmieniać, dodawać oraz usuwać. Ponadto, Pracownik może usunąć umówioną jazdę i w pojawiającym się oknie wpisać przyczynę.

3.3.5 Stajenny

1. Dostępne dane jak u Pracownika
2. Dostępne funkcjonalności jak u Pracownika

3.3.6 Kierownik

1. Przechowywane dane jak u Pracownika
2. Dostępne funkcjonalności jak u Pracownika i dodatkowe:
    a. Dodaj dzierżawę
    b. Dodaj lub usuń pracownika
    c. Dodaj lub usuń konia
    d. Przeglądaj wszystkie zadania
    e. Przypisz dzierżawę do konia
    f. Zmień stanowisko Pracownika

Ad.3.3.6.2.a/e – Kierownik ma możliwość przypisania konia do dzierżawy klientowi.

Ad.3.3.6.2.b – Kierownik dodaje i usuwa Pracowników z systemu.

Ad.3.3.6.2.c – Kierownik może dodawać i usuwać konie z systemu.

3.3.7 Kowal

1. Przechowywane dane jak u Pracownika
2. Dostępne funkcjonalności jak u Pracownika

3.3.8 Weterynarz

1. Przechowywane dane jak u Pracownika
2. Dostępne funkcjonalności jak u Pracownika i dodatkowe:
    a. Akceptacja lub odrzucenie zgłoszenia o problemie zdrowotnym otrzymanego od innego użytkownika systemu.
    b. Umówienie wizyty weterynaryjnej
    c. Podsumowanie wizyty weterynaryjnej

Ad.3.3.8.1.a - dla zaakceptowanych zgłoszeń może zmienić priorytet nadany przez zgłaszającego, „niski”, „średni” lub „wysoki” – na liście zgłoszeń są one posortowane od priorytetu wysokiego do niskiego, zadania bez priorytetu są w kolejce.

Ad.3.3.8.1.c - gdy Weterynarz przyjmie zgłoszenie i zbada konia może w systemie doprecyzować opis zgłoszenia, może dodać notatkę opisująca sposób leczenia oraz w grafiku karmienia dodać wskazówki lub wskazania odnośni karmienia konia w okresie rekonwalescencji.

3.3.9 Instruktor

1. Dostępne funkcjonalności jak u Pracownika i dodatkowo:
    a. Tworzenie jazd/lekcji w swoim grafiku
    b. Edytowanie jazd/lekcji w swoim grafiku
    c. Zmiana statusu lekcji
    d. Zmiana daty zajęć

Ad.3.3.9.1.a - Instruktor ma możliwość dodawania zajęć/jazd w swoim grafiku. Przy tworzeniu nowego wydarzenia musi wybrać datę oraz godzinę, ma możliwość zaznaczenia czy zajęcia te odbywać się będą cyklicznie, przez co nie musi wprowadzać kilkakrotnie tego samego zajęcia np. co tydzień. Instruktor musi podać maksymalną ilość uczestników tych zajęć oraz otagować, np. „dla początkujących".

Ad.3.3.9.1.b - Instruktor ma możliwość edytowania zajęć/jazd w swoim grafiku – może dodać konia do zajęć, na którym będą jeździć uczestnicy, może usunąć konia z zajęć, może dodawać i usuwać klientów, którzy wezmą udział w zajęciach, Instruktor może przypisać konia do Klienta na konkretne zajęcia oraz może zmienić datę i godzinę, kiedy zajęcia się odbędą.

Ad.3.3.9.1.c – Instruktor ma możliwość zmiany statusu lekcji, np. na odwołana lub zrealizowana.

3.4 Aplikacja mobilna

System SIDOS zakłada powstanie aplikacji mobilnej, która będzie posiadała mniej funkcjonalności niż aplikacja webowa.

3.4.1 Po stronie Gościa:

1. Dostępne funkcjonalności:
    a. Przeglądanie publicznych profili pracowników zatrudnionych przez ośrodek
    b. Przeglądanie profili koni
    c. Wyszukiwanie koni na podstawie cech charakteru

Ad.3.4.1.1.a – Gość ma pogląd na profile koni, na których będą wyświetlały się jego zdjęcie, imię oraz opis.

Ad.3.4.1.1.b - Gość będzie mógł oglądać profile Pracowników, na których będą wyświetlały się zdjęcie profilowe, imię, nazwisko, dane kontaktowe oraz jego grafik.

3.4.2 Po stronie Klienta

1. Dostępne funkcjonalności:
    a. Takie, jak u Gościa
    b. Przeglądanie swojego grafiku jazd
    c. Wysyłanie i odbieranie wiadomości
    d. Przeglądanie strony z ogłoszeniami

Ad.3.4.2.1.a – Klient może oglądać swój grafik z opcją szybkiego anulowania zajęć.

3.4.3 Po stronie Pracownika

1. Dostępne funkcjonalności:
    a. Odbieranie powiadomień
    b. Przeglądanie swojego grafiku
    c. Wysyłanie zgłoszeń do weterynarza
    d. Dodawanie i edytowanie ogłoszeń

3.4.4 Po stronie Weterynarza

1.Dostępne funkcjonalności:
    a. Takie same, jak w przypadku pracownika
    b. Przyjmowanie zgłoszeń o problemie zdrowotnym
    c. Odrzucanie zgłoszeń o problemie zdrowotnym
    d. Zmiana statusu zgłoszeń o problemie zdrowotnym

3.5 Wymagania funkcjonalne

SIDOS jest prototypem systemu do zarządzania ośrodkiem jeździeckim. W swoim założeniu system ma być elastyczny, dzięki czemu dopasuje się do zróżnicowanych wymagań klientów i potrzeb różnych stajni. Jego celem jest automatyzacja wielu procesów, co spowoduje usprawnienie komunikacji na poziomie pracowników i klientów oraz pozwoli na przechowywanie wszystkich danych dotyczących funkcjonowania stadniny w jednym miejscu. Informacje zebrane na wczesnym etapie analizy przekładają się na poniżej wymienione funkcjonalności systemu. Błędne określenie funkcji może spowodować, iż system będzie nieefektywny lub w gorszym przypadku bezużyteczny.

Wymagania funkcjonalne zebrane w tabelach określają funkcje, które musi posiadać system. Podstawowym warunkiem początkowym każdej funkcjonalności jest zalogowanie użytkownika do systemu.

Tabela 1 zawiera wymagania funkcjonalne dotyczące jazd.

Tabela 1. Wymagania funkcjonalne dotyczące jazd. Źródło: Opracowanie własne.

L.P.	Nazwa	Opis Wymagania	Aktorzy	Warunki początkowe
1.1	Umów się na jazdę	Aby umówić się na zajęcia Klient musi wejść w zakładkę „Umów się na zajęcia" dostępną w jego panelu. Następnie w poprzez kalendarz musi wybrać dzień, w którym chce wziąć udział w lekcji	Klient	W systemie utworzona jest co najmniej jedna jazda
1.2	Odwołaj jazdę	Poprzez przejście w zakładkę „Moje jazdy" klient może kliknąć „Odwołaj jazdę" przy jeździe, na którą jest zapisany.	Klient	Klient zapisany na co najmniej jedną jazdę
1.3	Wyświetl moje jazdy	Poprzez przejście w zakładkę "Moje jazdy " Klient może wyświetlić jazdy na, które został zapisany	Klient	W bazie istnieją jazdy
1.4	Dodaj jazdę	Instruktor ma możliwość dodawania jazd do swoich lekcji, które będą wyświetlane w jego grafiku. Przy tworzeniu nowego wydarzenia musi wybrać datę oraz godzinę	Instruktor	W bazie istnieje przynajmniej jedna lekcja
1.6	Edytuj jazdę	Instruktor posiada możliwość edycji jazdy poprzez wejście w zakładkę „Wyświetl moje lekcje" i kliknięcie „Wyświetl jazdy" a następnie „Edytuj jazdę". Instruktor posiada możliwość zmiany statusu jazdy.	Instruktor	Jazda musi być dodana do bazy danych

Tabela 2 opisuje wymagania funkcjonalne dotyczące lekcji.

Tabela 2. Wymagania funkcjonalne dotyczące lekcji. Źródło: Opracowanie własne.

L.P.	Nazwa	Opis Wymagania	Aktorzy	Warunki początkowe
2.1	Wyświetl moje lekcje	W danym panelu instruktor ma możliwość podglądu na zaplanowane lub odbyte lekcje	Instruktor	W bazie istnieją lekcje
2.2	Dodaj lekcję	Instruktor z menu wybiera przycisk "Dodaj lekcje" gdzie system wyświetla są pola dotyczące dodania nowej lekcji do systemu	Instruktor	
2.4	Edytuj lekcję	Edycja lekcji poprzez formularz	Instruktor	Lekcja musi być dodana do bazy danych

Tabela 3 przechowuje wymagania funkcjonalne dotyczące zadań.

Tabela 3. Wymagania funkcjonalne dotyczące zadań. Źródło: Opracowanie własne.

L.P.	Nazwa	Opis Wymagania	Aktorzy	Warunki początkowe
3.1	Dodaj zadanie	Użytkownik wybiera zakładkę "Dodaj zadanie" gdzie chce zlecić dane zadanie konkretnej osobie.	Kierownik	
3.2	Przeglądaj swoje zadania	Miejsce, w którym wyświetlane są zadania dla konkretnego pracownika	Kowal, Stajenny, Kierownik	
3.3	Przeglądaj wszystkie zadania	Panel służący wyłącznie kierownikowi stajni umożliwiający pogląd do aktualnych zadań pracowników znajdujących się pod zakładka "Przeglądaj wszystkie zadania"	Kierownik	W bazie musza istnieć zadania
3.4	Przeglądaj szczegóły zadania	Funkcjonalność pozwalająca na zapoznanie się ze szczegółami zadania.	Kowal, Stajenny, Kierownik	Zadnia istnieją w systemie
3.5	Zmień status zadania	Funkcjonalność pozwalająca oznaczyć zadanie jako zaakceptowane bądź wykonane.	Kowal, Stajenny, Kierownik	Zadanie musi istnieć w systemie
3.6	Edytuj zadanie	Kierownik posiada możliwość Edycji danego zadania	Kierownik	Zadanie musi istnieć w systemie
3.7	Usuń zadanie	Funkcjonalność dedykowana dla kierownika pozwalając usunąć zadanie z systemu.	Kierownik	Zadanie musi istnieć w systemie

Tabela 4 opisuje wymagania funkcjonalne dotyczące wizyt weterynaryjnych.

Tabela 4. Wymagania funkcjonalne dotyczące wizyt weterynaryjnych. Źródło: Opracowanie własne.

L.P.	Nazwa	Opis Wymagania	Aktorzy	Warunki początkowe
4.1	Wyświetl wizyty weterynaryjne	Pod postacią listy wyświetlane są aktualne wizyty weterynaryjne	Weterynarz, Kierownik, Właściciel, Dzierżawca	W bazie istnieją dodane wizyty weterynaryjne
4.2	Umów wizytę weterynaryjna	Właściciel oraz dzierżawca w razie potrzeby może zgłosić wizytę weterynaryjna dzięki sprecyzowanemu formularzowi	Kierownik, Weterynarz	
4.3	Przeglądaj historie wizyt weterynaryjnych	W danej zakładce wyświetlona jest historia poszczególnych wizyt weterynaryjnych danego konia z dokładna data oraz weterynarzem przeprowadzaj acyl wizytę	Właściciel, Weterynarz, Dzierżawca, Kierownik	W bazie widnieją wizyty, które się już odbyły
4.4	Edytuj wizytę weterynaryjną	Po przejściu do zakładki „Wyświetl wizyty weterynaryjne” istnieje możliwość edycji wizyty w celu nadania jej daty spotkania, symptomów choroby, komentarza oraz diagnozy.	Weterynarz, Kierownik	W bazie istnieje co najmniej jedna zaplanowana wizyta

Tabela 5 przechowuje wymagania funkcjonalne dotyczące problemu zdrowotnego.

Tabela 5. Wymagania funkcjonalne dotyczące problemu zdrowotnego. Źródło: Opracowanie własne.

L.P.	Nazwa	Opis Wymagania	Aktorzy	Warunki początkowe
5.1	Zgłoś problem zdrowotny	Pracownik ma możliwość wysłania zgłoszenia do weterynarza, gdy podejrzewa kontuzję lub uraz u konia. Przy wysyłania takiego zgłoszenia musi podać opis zdarzenia lub objawów	Użytkownik	W bazie istniała konie oraz weterynarze

5.2	Wyświetl zgłoszenia o problemie zdrowotnym	Strona wyświetla zgłoszenia o problemie zdrowotnym konia	Weterynarz, Kierownik	W bazie istnieją zgłoszenia o problemie zdrowotnym

Tabela 6 zawiera wymagania funkcjonalne dotyczące panelu kierownika.

Tabela 6. Wymagania funkcjonalne dotyczące panelu kierownika. Źródło: Opracowanie własne.

L.P.	Nazwa	Opis Wymagania	Aktorzy	Warunki początkowe
6.1	Dodaj pracownika	Kierownik posiada uprawnienia do dodawania pracowników do systemu, gdzie przechowywane są ich dane osobowe	Kierownik	
6.2	Dodaj konia	Jest to funkcjonalność, która odpowiada za dodawanie koni do bazy danych	Kierownik	
6.3	Dodaj właściciela konia	Jest to funkcjonalność, która odpowiada za dodawanie właścicieli koni do bazy danych	Kierownik	
6.4	Wszyscy pracownicy	Kierownik ma możliwość sprawdzenia danych wszystkich pracowników	Kierownik	W systemie istnieje przynamniej jeden pracownik
6.5	Wszyscy właściciele koni	Kierownik ma możliwość sprawdzenia danych wszystkich właścicieli koni	Kierownik	W systemie istnieje przynamniej jeden właściciel
6.6	Wszystkie konie	Kierownik ma możliwość sprawdzenia danych wszystkich koni	Kierownik	W systemie istnieje przynamniej jeden koń

Tabela 7 opisuje wymagania funkcjonalne dotyczące dzierżawy konia.

Tabela 7. Wymagania funkcjonalne dotyczące dzierżawy konia. Źródło: Opracowanie własne.

L.P.	Nazwa	Opis Wymagania	Aktorzy	Warunki początkowe
7.1	Dodaj dzierżawę	Poprzez wejście w zakładkę „dodaj dzierżawę" można przypisać konia i użytkownika systemu do dzierżawy	Kierownik	W systemie jest co najmniej jeden koń dostępny do dzierżawy i jeden użytkownik
7.2	Przeglądaj dzierżawy	Poprzez wejście w zakładkę „przeglądaj dzierżawy" istnieje możliwość podglądu wszystkich aktualnych dzierżaw	Kierownik	W systemie istnieje co najmniej jedna dzierżawa
7.3	Edytuj dzierżawę	Istnieje możliwość edycji dzierżawy poprzez wejście w zakładkę „Przeglądaj dzierżawy" i kliknięcie „edytuj"	Kierownik	W systemie istnieje co najmniej jedna dzierżawa
7.4	Szczegóły dzierżawy	Istnieje możliwość sprawdzenia szczegółów dzierżawy poprzez wejście w widok „Przeglądaj dzierżawy" a następnie kliknięcie w „Szczegóły"	Kierownik	W systemie istnieje co najmniej jedna dzierżawa

Tabela 8 przedstawia wymagania funkcjonalne dotyczące zarządzania profilem.

Tabela 8. Wymagania funkcjonalne dotyczące zarządzania profilem

L.P.	Nazwa	Opis Wymagania	Aktorzy	Warunki początkowe
8.1	Zmień hasło	Klient może zmienić swoje hasło (w dowolnym wybranym przez siebie momencie, minimum pół roku po ostatniej zmianie hasła	Klient	

8.2	Zmień zdjęcie profilowe	Użytkownik ma możliwość edycji profilu poprzez wejście w zakładkę „Mój profil”	Klient	
8.3	Edytuj swoją galerię zdjęć	Możliwość dodawania oraz usuwania zdjęć.	Instruktor	
8.4	Certyfikaty	Poprzez zakładkę Certyfikaty Instruktor może dodać, usuwać i edytować certyfikaty	Instruktor	
8.5	Zmień dane profilowe	Edycja danych następująca poprzez przejście w zakładkę zarządzaj profilem (imię, nazwisko, zdjęcie, data urodzenia.)	Klient	
8.6	Przeglądaj kalendarz	W zależności od zalogowanego użytkownika w kalendarzu prezentowane są różne zdarzenia – lekcje, zadania, wizyty weterynaryjne	Klient, Pracownik, Kierownik	

Tabela 9 prezentuje wymagania funkcjonalne dotyczące strony startowej oraz sekcji z podstawowymi informacjami o stajni.

Tabela 9. Wymagania funkcjonalne dotyczące strony startowej oraz sekcji z podstawowymi informacjami o stajni. Źródło: Opracowanie własne.

L.P.	Nazwa	Opis Wymagania	Aktorzy	Warunki początkowe
9.1	Logowanie	Każda osoba zainteresowana stadnina koni ma możliwość zalogowania się do swojego prywatnego profilu w celu uzyskania dostępu do funkcjonalności, jakie oferuje system oraz uzyskać wgląd do bieżących informacji o stadninie oraz komunikacji z innymi użytkownikami portalu.	Wszyscy	Użytkownik zarejestrowany do systemu.
9.2	Rejestracja	Osoba zainteresowana stadnina koni ma możliwość rejestracji swojego profilu w celu zwiększenia	Wszyscy	Brak

		funkcjonalności i posiadania profilu użytkownika.		
9.3	Przeglądaj aktualności	Widok, na którym znajdują się najnowsze aktualności dotyczące wydarzeń oraz informacji dotyczących stadniny.	Wszyscy	Do bazy muszą a być dodane aktualności
9.4	Dodaj aktualności	Kierownik jest odpowiedzialny za dodawanie aktualności które wyświetlane są na stronie startowej. Aktualności zawierają datę, zdjęcie oraz opis	Kierownik	Brak
9.5	Edytuj aktualności	Przycisk „Edytuj” dedykowany dla kierownika pozwala na edycje bieżących informacji w konkretnej aktualności	Kierownik	Do bazy muszą a być dodane aktualności
9.6	Przeglądaj profile pracowników	Widok z lista pracowników stadniny przedstawiający ich opis oraz wykonywany zawód.	Wszyscy	W bazie musza istnieć pracownicy
9.7	Przeglądaj profile koni	Widok prezentujący listę koni wraz opisem i ikoną charakteru konia, która jest inny dla każdego	Wszyscy	W bazie musza istnieć konie
9.8	Przeglądaj dane kontaktowe	Widok, w którym przedstawione są wszystkie niezbędne dane kontaktowe wraz z mapa lokalizacji stadniny	Wszyscy	Brak

Tabela 10 przedstawia funkcjonalności związane z wysyłaniem i odbieraniem wiadomości w obrębie systemu.

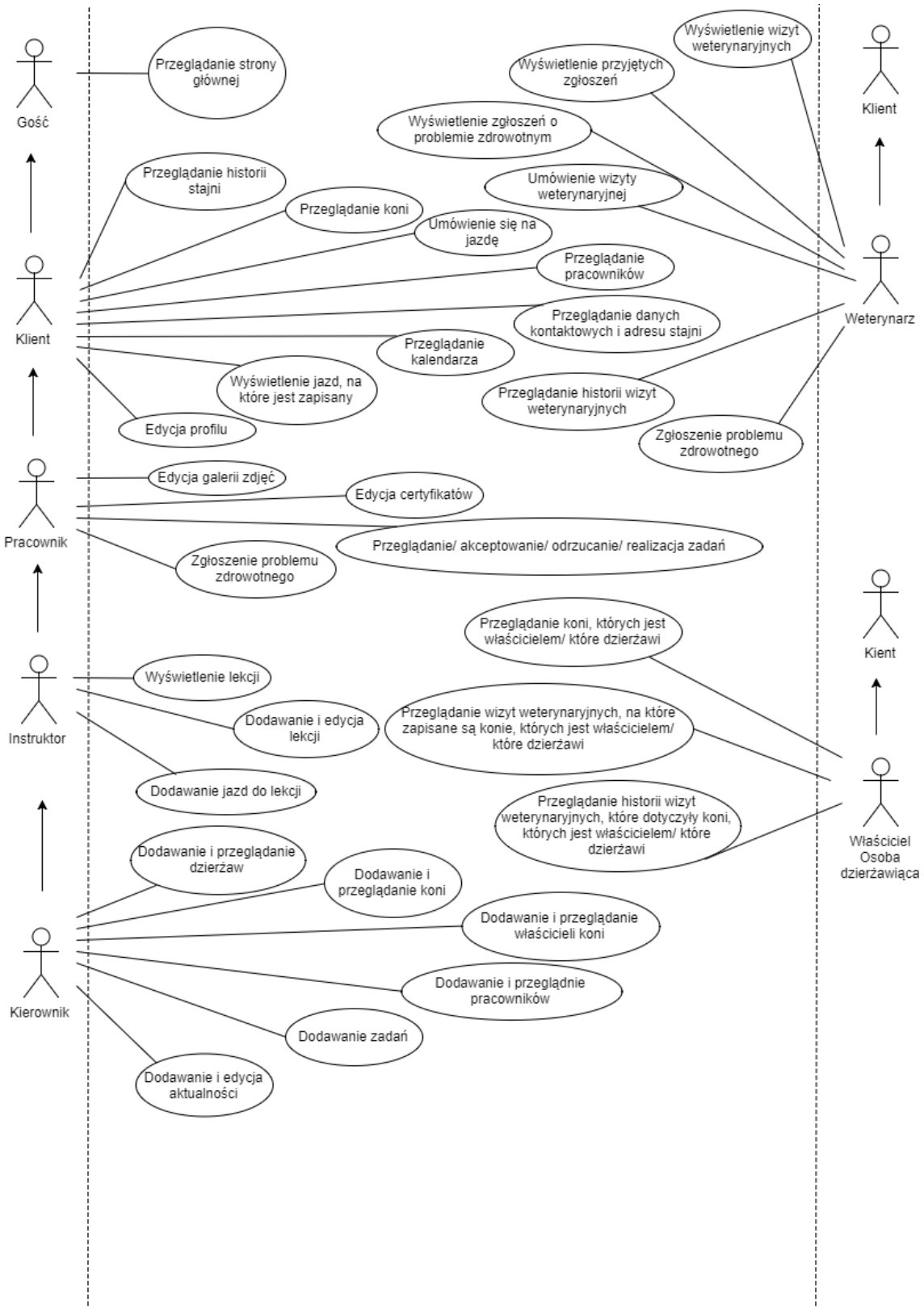
Tabela 10. Funkcjonalności związane z wiadomościami.

L.P.	Nazwa	Opis Wymagania	Aktorzy	Warunki początkowe
10.1	Nowa wiadomość	Istnieje możliwość wysłania wiadomości do osób zarejestrowanych w systemie	Klient	

10.2	Skrzynka odbiorcza	Istnieje możliwość sprawdzenia podstawowych informacji o wiadomości	Klient	
10.3	Skrzynka nadawcza	Istnieje możliwość sprawdzenia podstawowych informacji o wysłanych wiadomościach	Klient	
10.4	Odczytaj wiadomość	Poprzez wejście w szczegóły wiadomości (ze skrzynki odbiorczej oraz nadawczej) można odczytać jej treść, sprawdzić nadawcę, odbiorcę oraz datę wysłania	Klient	W systemie istnieje co najmniej jedna wiadomość

3.6 Przypadki użycia

Na rysunku 13 zaprezentowano diagram przypadków użycia.



Rysunek 13. Przypadki użycia w systemie SIDOS.

3.7 Wymagania niefunkcjonalne

Wymagania niefunkcjonalne przedstawiają właściwości, które system powinien realizować oraz ograniczenia związane z wydajnością, bezpieczeństwem, użytecznością, niezawodnością. Wymagania nie dotyczą bezpośrednio funkcji biznesowych natomiast są związane z czynnikami dotyczącymi oprogramowania, sprzętu lub potrzeb użytkownika. Wymagania niefunkcjonalne zostały przedstawione w tabeli 11.

Tabela 11. Wymagania niefunkcjonalne. Źródło: Opracowanie własne

L.p.	**Cecha**	**Priorytet**	**Opis**
1	Użyteczność	Powinien	System obsługiwany w przeglądarce w języku polskim
2	Użyteczność	Wymagany	Odpowiednia przeglądarka internetowa (Internet Explorer 11, Opera 50, Chrome 63, Firefox 57 i w wersjach powyżej)
3	Wydajność	Wymagany	Serwer przesyła stronę w czasie poniżej 10 sekund
4	Wydajność	Wymagany	Serwer obsługuje 100 użytkowników jednocześnie
5	Użyteczność	Wymagany	System działa na systemach operacyjnych Windows
6	Użyteczność	Wymagany	Aplikacja mobilna działa w systemie operacyjnym Android
7	Bezpieczeństwo	Powinien	Hasła przechowywane w formie zaszyfrowanej
8	Niezawodność	Wymagany	Wymagana ilość pamięci RAM 2GB
9	Niezawodność	Powinien	Wymagana ilość pamięci dyskowej 2GB

4 Opis narzędzi i technologii użytych w pracy

W rozdziale tym zostały opisane narzędzia i technologie użyte w pracy.

4.1 Platforma .NET

.NET to projekt firmy Microsoft rozpoczęty w późnych latach 90, obejmujący opracowanie nowej technologii programistycznej, który znacząco podniósł stan współczesnego doświadczenia w zakresie implementacji języków programowania. Kluczowym elementem było wsparcie dla wielu języków programowania z jednym językiem uruchomieniowym (tzw. Common Language Runtime – CLR), ale obejmował także wiele innych mniejszych dodatków, takich jak typy wartości, prosty model wyjątków i atrybuty. Zintegrowane kwerendy generyczne i językowe zostały później dodane do tej listy. Objęcie powyższych obszarów standaryzacjami ECMA oraz ISO, doprowadziło do aktualnej popularności tego rozwiązania programistycznego. Jedną z najważniejszych i najpopularniejszych implementacji technologii .NET jest NET Framework.

Obecnie ponad 40 [10] języków programowania jest zgodnych z .NET. Do podstawowych dostarczanych przez Microsoft należą: C#, Visual Basic .NET, F#, C++/CLI, J# (wariant języka Java opracowany przez Microsoft), JScript .NET (kompilowany wariant języka JScript).

Inne języki programowania zgodne z platformą .NET to m.in.: COBOL, Delphi (Delphi.NET od wersji 8 środowiska), Fortran, Lisp, Perl, Python czy Smalltalk.

W celu umożliwienia oddzielenia języka programowania od środowiska uruchomieniowego firma Microsoft opracowała otwartą specyfikacją CLI (Common Language Infrastructure) opisaną w standardzie ECMA-335.

Standard ten definiuje wspólną infrastrukturę językową (CLI), w której aplikacje napisane w różnych językach wysokiego poziomu mogą być wykonywane w różnych środowiskach systemowych bez konieczności przepisywania tych aplikacji w celu uwzględnienia unikalnych właściwości tych środowisk. Powyższy standard składa się z następujących części:

1. Koncepcje i architektura - opisuje ogólną architekturę interfejsu CLI i dostarcza normatywny opis systemu Common Type (CTS), systemu Virtual Execution System

(VES) i specyfikacji języka wspólnego (CLS). Zapewnia także informacyjny opis metadanych.

2. Definicja metadanych i semantyka - zapewnia normatywny opis metadanych: jego fizyczny układ (jako format pliku), jego logiczną treść (jako zbiór tabel i ich powiązania) i jego semantykę (widzianą z hipotetycznego asemblera ilasta).
3. CIL Lista instrukcji - opisuje zestaw instrukcji CIL (Common Intermediate Language).
4. Profile i biblioteki - zawiera przegląd bibliotek CLI i specyfikację ich faktoringu w profile i biblioteki. Plik towarzyszący CLILibrary.xml, uważany za część tej partycji, ale dystrybuowany w formacie XML, zawiera szczegółowe informacje o każdej klasie, typie wartości i interfejsie w bibliotekach CLI.
5. Format wymiany debugowania - opisuje standardowy sposób wymiany informacji dotyczących debugowania między producentami CLI i konsumentami.
6. Załączniki - zawiera niektóre przykładowe programy napisane w języku CIL Assembly Language (ILAsm), informacje o konkretnej implementacji asemblera, odczytywalny maszynowo opis zestawu instrukcji CIL, który może być użyty do wyprowadzenia części gramatyki stosowanej w niniejszym asembler, a także inne narzędzia, które manipulują CIL, zestaw wytycznych używanych w projektowaniu bibliotek partycji IV oraz rozważania dotyczące przenośności. [11]

Odpowiednikiem powyższych standardów w języku Java są JVM (Java Virtual Machine) dla CLR, oraz SDK (Java Software Development Kit) dla CLI.

.NET Framework (w skrócie .NET) to technologia programistyczna opracowana przez firmę Microsoft. Zawiera bibliotekę klas o nazwie Framework Class Library (FCL) i zapewnia interoperacyjność językową (każdy język może wykorzystywać kod napisany w innych językach) w wielu językach programowania. Programy napisane dla platformy .NET Framework są uruchamiane w środowisku o nazwie Common Language Runtime (CLR), wirtualnej maszynie aplikacji, która zapewnia takie usługi, jak bezpieczeństwo, zarządzanie pamięcią i obsługa wyjątków. FCL i CLR razem stanowią .NET Framework.

FCL zapewnia interfejs użytkownika, dostęp do danych, łączność z bazą danych, kryptografię, tworzenie aplikacji internetowych, algorytmy numeryczne i komunikację sieciową. Programiści tworzą oprogramowanie, łącząc kod źródłowy z .NET Framework i innymi bibliotekami. Framework jest przeznaczony do użycia przez większość nowych

aplikacji tworzonych na platformę Windows. Microsoft tworzy także zintegrowane środowisko programistyczne głównie dla oprogramowania .NET o nazwie Visual Studio.

Microsoft rozpoczął rozwój .NET Framework pod koniec lat 90, początkowo pod nazwą Next Generation Windows Services (NGWS). Pod koniec 2000 roku zostały wydane pierwsze wersje beta .NET 1.0. Aktualna wersja to 4.7.1 wydana 17.10.2017 r.

Inne alternatywy dla .NET Framework to m.in.:

- .NET Micro Framework – jest przeznaczony do środowisk o bardzo ograniczonych zasobach,
- .NET Core – alternatywna implementacja, której najważniejszymi cechami jest wielo-platformowość (Windows, macOS i GNU/Linuks) oraz licencja Open Source,
- Mono – to implementacja CLI oraz FCL, o podwójnej licencji (darmowej oraz ograniczonej), przeznaczona dla różnych systemów operacyjnych (np. Android, Windows Phone, iOS, GNU/Linuks, BSD, OS X, Windows, Solaris).

4.2 Wzorzec projektowy Model-View-Controller

MVC (Model-View-Controller) jest jednym z podstawowych wzorców architektonicznych zorientowanych obiektowo, wykorzystywany w tworzeniu oprogramowania. Wzorzec ten został wykorzystany głównie dlatego, że ułatwia podział obowiązków w implementacji poprzez oddzielenie warstwy prezentacji od logiki biznesowej aplikacji.

Wzorzec MVC został zaprojektowany w 1979 roku, podczas prac na językiem Smalltalk i z biegiem czasu ewoluował do różnych wariantów. Wzorzec ten dzieli się na trzy warstwy:

- Model (Model) – w nim znajdują się klasy implementujące logikę biznesową aplikacji (w niektórych przypadkach również w serwisach) oraz sposób przechowywania danych (za pośrednictwem wzorca projektowego Repozytorium [12]).
- View (Widok) – warstwa prezentacji. Udostępnia użytkownikowi fuknjonalności aplikacji, do których ma uprawnienia.

– Controller (Kontroler) – warstwa, która pośredniczy pomiędzy modelem a widokiem. Przechwytuje z widoku akcje użytkownika, na ich podstawie aktualizuje model i odświeża widoki.

Zalety:

– Jednoczesny rozwój - wiele osób może pracować jednocześnie nad modelem, kontrolerem i widokami.
– Spójność - MVC umożliwia logiczne grupowanie powiązanych działań na kontrolerze. Widoki dla konkretnego modelu są również pogrupowane.
– Luźne powiązania (*loose coupling*) - sama natura MVC jest taka, że występuje małe powiązanie pomiędzy modelami, widokami lub kontrolerami.
– Łatwość modyfikacji - ze względu na podział obowiązków, rozwój lub modyfikacja aplikacji jest łatwiejszy.

Niedogodności:

– Złożoność kodu - nawigacja po strukturze może być skomplikowana, gdyż wprowadza nowe warstwy abstrakcji i wymaga od użytkowników dostosowania się do kryteriów dekompozycji MVC.
– Konsolidacja wielu artefaktów - dekompozycja obiektu na trzy artefakty powoduje rozproszenie. Wymagało to od programistów zachowania spójności wielu reprezentacji jednocześnie.
– Krzywa uczenia się - wiedza na temat wielu technologii staje się normą. Programiści korzystający z MVC muszą być wykwalifikowani w wielu technologiach.

4.3 Microsoft Visual Studio 2015

Microsoft Visual Studio [13] to zintegrowane środowisko programistyczne (Integrated Development Environment - IDE) firmy Microsoft. Służy do tworzenia oprogramowania komputerowego, stron internetowych, aplikacji internetowych, usług internetowych i aplikacji mobilnych. Visual Studio wykorzystuje platformy programistyczne firmy Microsoft, takie jak Windows API, Windows Forms, Windows Presentation Foundation, Windows Store i Microsoft Silverlight. Może produkować zarówno kod natywny (np. dla języka C++), jak i kod zarządzany (np. dla języka C#).

Visual Studio zawiera edytor kodu obsługujący IntelliSense (komponent uzupełniania kodu) oraz refaktoryzację kodu. Zintegrowany debugger działa zarówno jako debugger na poziomie źródłowym, jak i debugger na poziomie komputera. Inne wbudowane narzędzia obejmują profiler kodu, projektant formularzy do budowania aplikacji GUI, projektant stron internetowych, klas i schematów baz danych. Do VS można instalować pluginy (wtyczki), które zwiększają funkcjonalność wielu poziomach - w tym m.in. dodawanie obsługi systemów kontroli wersji kodu źródłowego (np. Subversion, Git), nowych zestawów narzędzi, takich jak edytory czy środowiska do graficznego modelowana specyficznego dla danej domeny lub zestawów narzędzi dla innych aspektów cyklu rozwoju oprogramowania (jak Team Explorer, Properties Editor, Data Explorer).

Visual Studio obsługuje wiele różnych języków programowania i umożliwia edytorowi kodu i debuggerowi obsługę (w różnym stopniu) prawie dowolnego języka programowania, pod warunkiem, że istnieje usługa specyficzna dla języka. Języki wbudowane to C, C++, C++/CLI, Visual Basic .NET, C #, F #, JavaScript, TypeScript, XML, XSLT, HTML i CSS. Obsługa innych języków, takich jak Python, Ruby, Node.js i M jest dostępna za pośrednictwem wtyczek. Java (i J #) były obsługiwane w przeszłości.

Obecnie najnowsza wersja to Visual Studio 2017, jednak w tym rozdziale omówiona jest wersja wykorzystywana w pracy inżynierskiej – Visual Studio 2015, opublikowana 20 lipca 2015 roku.

Ogólny opis platformy

Visual Studio można wykorzystywać do budowania aplikacji konsolowych, webowych, mobilnych oraz serwisów webowych. Platforma udostępnia wiele funkcjonalności, w tym m.in.:

- Windows Forms Designer – tworzenie GUI aplikacji z użyciem Windows Forms.
- Windows Presentation Foundation (WPF) Designer – tworzenie GUI dla aplikacji desktopowych, lub umieszczonych wewnątrz obiektów na stronie www.
- Web designer/development – tworzenie dynamicznych stron www (np. ASP.NET), w tym m.in. z możliwością skorzystania z graficznego interfejsu typu drag’n’drop, do rozmieszczania na stronie kontrolek.
- Class designer – edytor klas, z możliwością wykorzystania notacji UML.

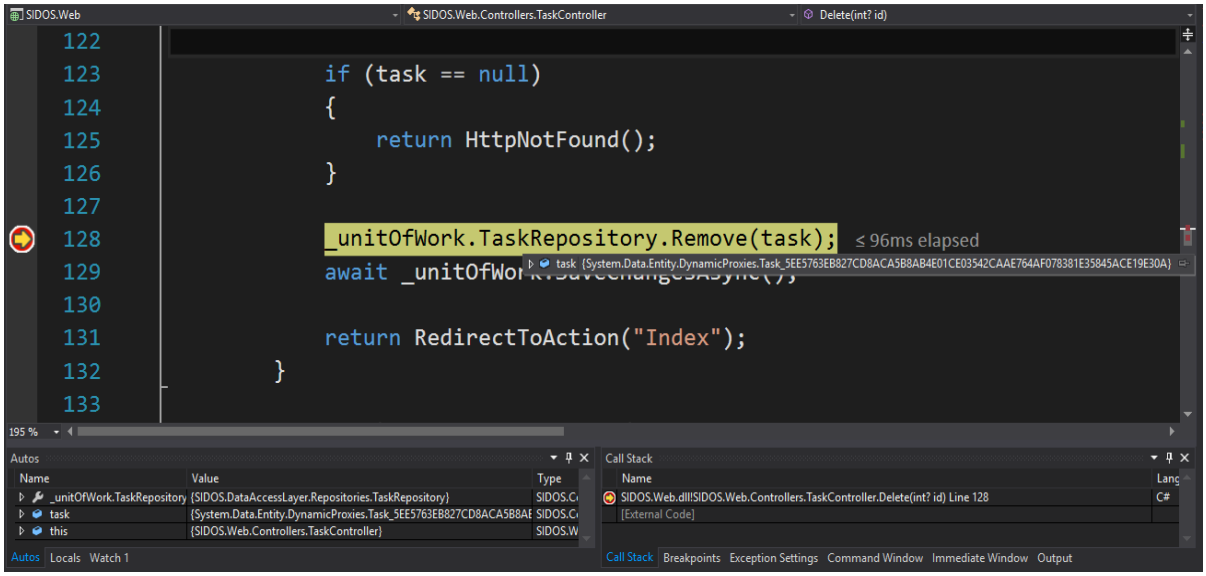
- Data designer – graficzne tworzenie i edycja schematów baz danych, oraz projektowanie zapytań SQL,
- Mapping designer – pozwala na mapowanie schematów baz danych oraz klas pozwalających na enkapsulację danych.
- Autouzupełnianie, informacja o parametrach, szybkie informacje i uzupełnianie słów w kodzie (ang. IntelliSense).
- Wbudowany menedżer pakietów NuGet [14]

W projekcie SIDOS wykorzystano dodatkowo pluginy m.in. do:

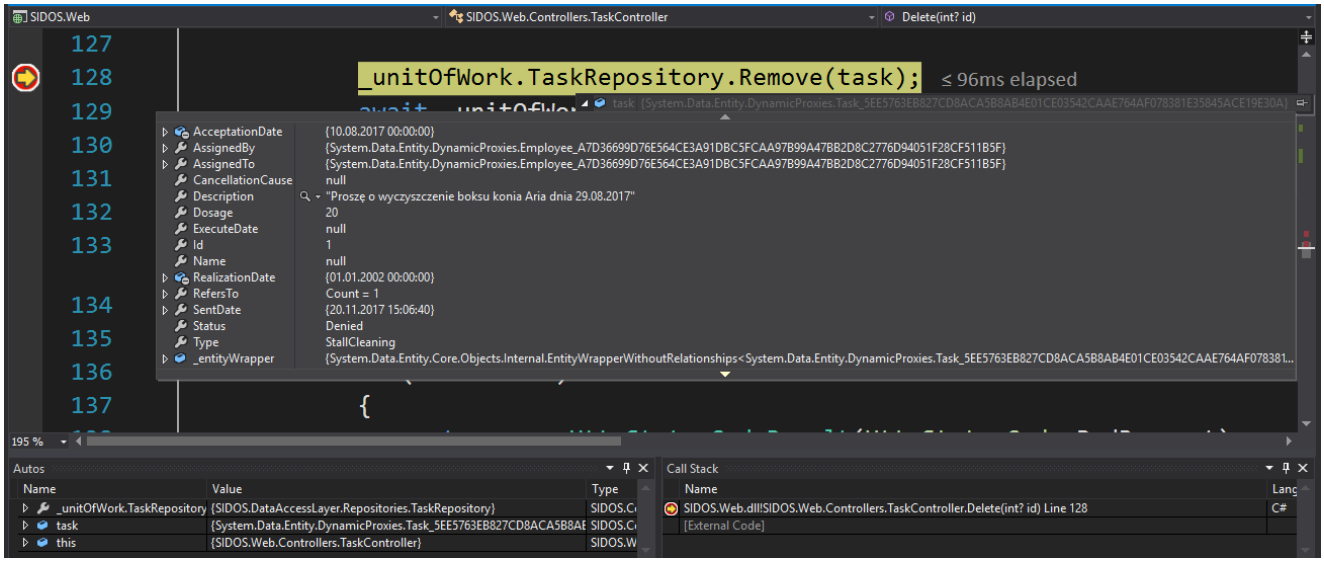
- Wersjonowania kodu (github);
- Usuwania niepotrzebnych przestrzeni nazw (Power Productivity Tools);
- Zmiany szaty wizualnej Visual Studio (Power Productivity Tools).

W projekcie korzystano z wielu funkcjonalności dostępnych w Visual Studio, np.

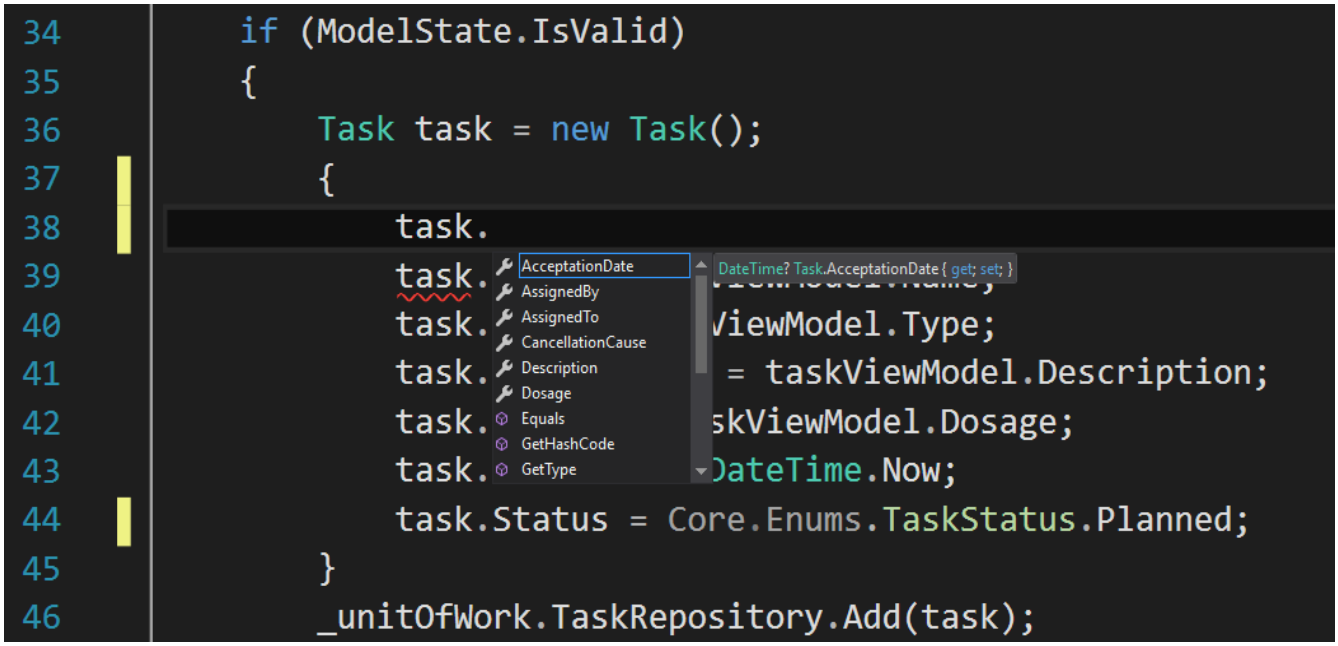
- Podziału projektu na podprojekty. Wydzielono z projektu części Core, Tests, DataAccessLayer i Web, dzieląc go na mniejsze, prostsze w utrzymaniu i korzystaniu elementy.
- Debugger. Korzystać z niego można na poziomie kodu źródłowego i maszyny. W trakcie działania programu (ang. Run Time) w środowisku testowym, jakim jest debugger, można m.in. podejrzeć wartości zmiennych w danym czasie, co przedstawia Rysunek 14 i Rysunek 15, lub zatrzymać wykonywanie kodu w określonych wcześniej punktach.
- Autoformatowania i kompilacji kodu w plikach .cshtml. Szczególnie w przypadku tagów HTML jest to użyteczna cecha, ponieważ cały kod jest automatycznie formatowany i nie ma potrzeby ręcznego poprawiania całego programu w celu poprawienia czytelności po dodaniu lub usunięciu tagu.
- Wspomaganie edycji kodu w postaci *IntelliSense.* W trakcie pisania programu możemy korzystać z sugerowanych poprawek w kodzie (Smart Fix) np. wyeliminować błąd w postaci powtarzających się nazw zmiennych, usunąć niewykorzystane przestrzenie nazw, zaimplementować automatycznie puste metody dziedziczonego interfejsu czy też przy pomocy listy rozwijanej wybierać dostępne metody i atrybuty dla stworzonego wcześniej obiektu. Rysunek 16 pokazuje przykład podpowiedzi w postaci listy obiektów, które są dostępne dla klasy Task.



Rysunek 14. Przykład sprawdzenia wartości zmiennej przy użyciu debuggera



Rysunek 15. Przykład sprawdzenia wartości zmiennych danego obiektu po rozwinięciu listy



Rysunek 16. Przykład działania IntelliSense. W tym wypadku widać podpowiedź w postaci listy rozwijanej, która pokazuje dostępne pola dla klasy Task.

4.4 Android Studio

SIDOS jest aplikacją dostępną na dwóch platformach – przeglądarkowo oraz jako aplikacja mobilna na telefonach wyposażonych w system operacyjny Android.

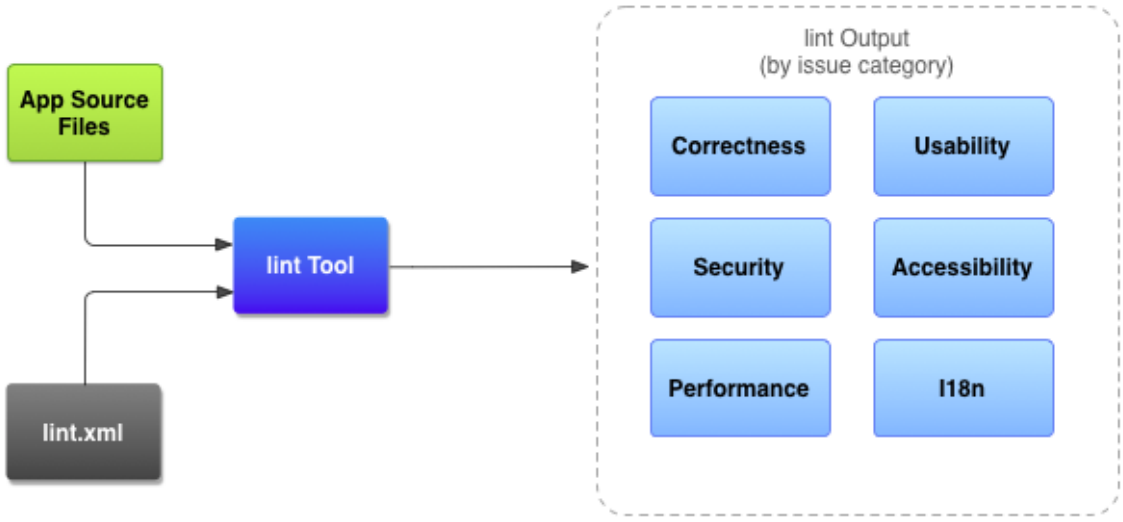
Android Studio jest aktualnie jedynym oficjalnym IDE dla rozwoju aplikacji systemu Android, opartym w budowie i UI na IntelliJ IDEA. Pod koniec 2014 roku zastąpił ówczesne IDE Eclipse Android Development Tools. Jest to narzędzie oferujące programiście m.in. szybki i bogaty w funkcje emulator umożliwiający testowanie manualne tworzonych aplikacji, integrację z systemem kontroli wersji Git, narzędzia monitorujące działanie i wydajność aplikacji (zużycie CPU, alokowane obiekty, przecieki pamięci, optymalizacja grafiki, żądania sieciowe).

Jako podstawę do budowania aplikacji Android Studio używa systemu Gradle [15], który działa jako zintegrowane narzędzie w menu Android Studio lub uruchamiany jest niezależnie z wiersza poleceń.

LINT

Android Studio oferuje swoim użytkownikom narzędzie do skanowania kodu Lint [16], które pomaga identyfikować i poprawiać błędy związane z jakością strukturalną kodu bez potrzeby uruchamiania aplikacji lub pisania testów. Każdy z problemów wykrytych przez Lint jest zgłaszany wraz z wiadomością opisującą rodzaj problemu. Dzięki temu programista może szybko ocenić priorytety problemów i zająć się ich naprawianiem, zaczynając od tych najbardziej krytycznych. Istnieje też opcja manualnego ustawiania priorytetów dla problemów, aby np. ignorować błędy niewpływające na projekt lub zwiększyć priorytet i podkreślić konkretne problemy.

Narzędzie Lint sprawdza pliki źródłowe projektu, by wykryć potencjalne błędy i poinformować o poprawkach optymalizacyjnych dla Poprawności (ang. correctness), Bezpieczeństwa (ang. security), Wydajności (ang. performance), Użyteczności (ang. usability) oraz Dostępności (ang. accessibility). Rysunek 17 przedstawia w uproszczony sposób działania narzędzia Lint.



Rysunek 17. Graficzne przedstawienie przetwarzania plików źródłowych przez narzędzie Lint

Podczas używania Android Studio, skonfigurowany Lint i inspekcje IDE uruchamiają się za każdym razem, kiedy buduje się aplikacja. Niemniej jednak programista ma możliwość ręcznego uruchomienia inspekcji z linii poleceń przy pomocy następującej komendy:

```
lint [ flags ] <project directory>
```

Dodatkowo można zażądać zeskanowania plików w folderze projektu i jego podfolderach. ID problemu MissingPrefix każe Lintowi szukać brakujących atrybutów XML, które nie mają prefiksu Android namespace. Komenda skanująca pliki w folderze myProject wygląda następująco:

```
lint –check MissingPrefix myProject
```

Rysunek 18 pokazuje informacje z konsoli, po uruchomieniu Lint w projekcie o nazwie Earthquake.

```
$ lint Earthquake

Scanning Earthquake: ...........................................................................................
Scanning Earthquake (Phase 2): .......
AndroidManifest.xml:23: Warning: <uses-sdk> tag appears after <application> tag [ManifestOrder]
  <uses-sdk android:minSdkVersion="7" />
  ^
AndroidManifest.xml:23: Warning: <uses-sdk> tag should specify a target API level (the highest verified version;
  <uses-sdk android:minSdkVersion="7" />
  ^
res/layout/preferences.xml: Warning: The resource R.layout.preferences appears to be unused [UnusedResources]
res: Warning: Missing density variation folders in res: drawable-xhdpi [IconMissingDensityFolder]
0 errors, 4 warnings
```

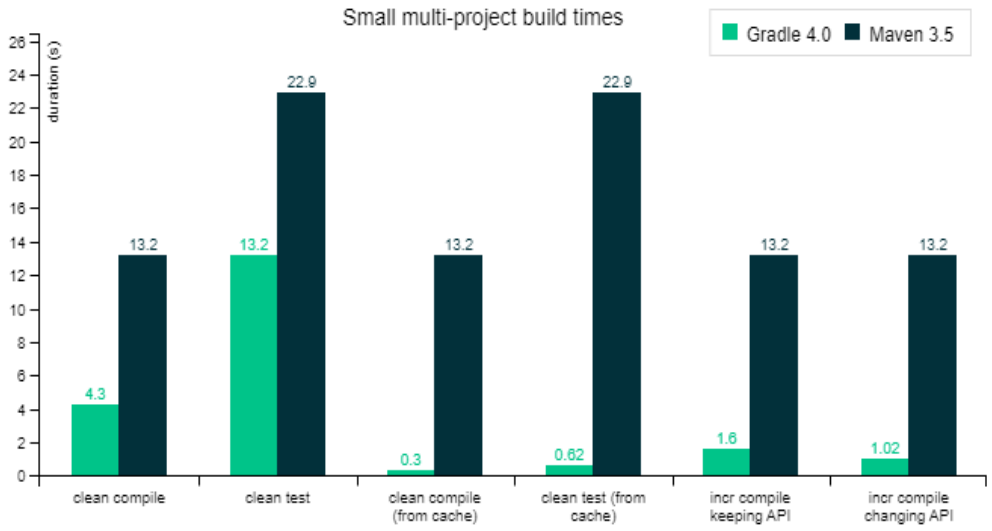
Rysunek 18. Przedstawienie informacji w konsoli po uruchomieniu Lint w projekcie

Gradle

Gradle to open-source'owy system budowania aplikacji, który umożliwia m.in.:

- Personalizację i konfigurację procesu budowania
- Tworzenie wielu APK[2] dla swojej aplikacji z różnymi cechami i właściwościami, używając tego samego projektu i modułów.
- Ponowne korzystanie z gotowego kodu

Alternatywą na rynku są takie systemy jak Apache Ant, Maven, CMake, Buildr, MSBuild, Rake, Continuum, Grunt. Jednak Gradle w przypadku małych i średnich projektów jest uznawany za najszybszy, co można zauważyć po przeprowadzonych badaniach. Rysunek 19 przedstawia wyniki badań.



Rysunek 19. Wyniki porównawcze dla systemów budowania aplikacji [17]

Po wydaniu polecenia budowania aplikacji poprzez wbudowany skrypt lub ręcznie, z linii wiersza poleceń, programista otrzymuje link, który przekierowuje go do strony Gradle, gdzie prezentowane są czasy wykonania i inne dane statystyczne.

[2] Android Package Kit – plik dystrybucyjny aplikacji dla systemu Android

4.5 Język programowania C#

C# jest wieloparadygmatowym, obiektowym językiem programowania, zaprojektowanym w latach 1998-2001 przez zespół firmy Microsoft. Głównymi autorami byli Anders Hejlsberg (twórca m.in. kompilatora Turbo Pascala firmy Borland, a w 2012r. języka TypeScript), Scott Wiltamuth i Peter Golde.

Nazwa i konstrukcja C# została zainspirowana językiem C, na co wskazuje m.in. pierwotna jego nazwa „C-like Object Oriented Language". C# jest zaakceptowanym oficjalnym językiem programowania opisanym w normach ECMA-334 oraz ISO/IEC 23270:2006. Aktualnie najnowszą wersją jest 7.2 wydana w 2017r.

C# kładzie nacisk na silne typowanie, imperatywne, deklaratywne, funkcjonalne, generyczne, obiektowe i komponentowe dziedziny programowania.

ECMA-334 opisuje główne cele C#:

1. Język ma być prostym, nowoczesnym, zorientowanym obiektowo językiem programowania ogólnego przeznaczenia.
2. Język i jego implementacje powinny zapewniać wsparcie dla zasad inżynierii oprogramowania, takich jak silne sprawdzanie typów, sprawdzanie granic tablic, wykrywanie prób użycia niezainicjowanych zmiennych i automatyczne usuwanie śmieci („garbage collection"). Ważna jest wytrzymałość oprogramowania, trwałość i wydajność programisty.
3. Język jest przeznaczony do używania w tworzeniu komponentów oprogramowania odpowiednich do wdrożenia w środowiskach rozproszonych.
4. Przenośność jest bardzo ważna dla kodu źródłowego i programistów, zwłaszcza tych, którzy już znają C i C ++.
5. Wsparcie dla internacjonalizacji jest bardzo ważne.
6. C # jest przeznaczony do pisania aplikacji zarówno na systemy hostowane, jak i osadzone, od bardzo dużych, które wykorzystują zaawansowane systemy operacyjne, aż po bardzo małe, posiadające dedykowane funkcje.
7. Chociaż aplikacje C# mają być ekonomiczne pod względem wymagań pamięci i mocy obliczeniowej, język nie miał na celu bezpośredniej rywalizacji o wydajność i rozmiar z C lub językiem asemblerowym [18].

C# jest zgodny z otwartą specyfikacją CLI (ECMA-334), co oznacza to, że zgodnie z powyższą normą, zaimplementowany jest wspólny zestaw typów danych (Common Type System - CTS) oraz wspólna specyfikacja języków (Common Language Specification – CLS). Po skompilowaniu program uruchamiany jest przez maszynę wirtualną firmy Microsoft CLR (Common Language Runtime). Dzięki zgodności z CLI, programy napisane w C#, mogą być uruchamiane na różnych platformach sprzętowych oraz środowiskach uruchomieniowych.

Odpowiednikiem powyższych w języku Java są JVM (Java Virtual Machine) dla CLR, oraz SDK (Java Software Development Kit) dla CLI.

C# jest językiem programowania opartym na językach C, C++ oraz Java, w związku z czym ma podobną składnię i zasady. Wspiera on przede wszystkim programowanie obiektowe, pozwala na programowanie funkcyjne, komponentowe oraz imperatywne.

C# cechuje się jednolitym systemem typów (Unified Type System) nazwanym tutaj Common Type System (CTS). Oznacza to, że wszystkie typy w tym języku dziedziczą z pojedynczego typu obiektu głównego.[19]

Język C#, podobnie jak Java, jest językiem wieloplatformowym –napisana aplikacja powinna funkcjonować na urządzeniach wykorzystujących różne systemy operacyjne. Język C# jest częścią platformy .NET, która pozwala na tworzenie oprogramowania w takich językach, jak Visual C++, Visual C#, Visual Basic, JScript.

Identyfikatory i słowa kluczowe

Identyfikator to słowo reprezentujące określone dane, np. klasy, metody, zmienne itd, mające charakter słowa. Przyjęło się, że zmienne lokalne, pola prywatne i parametry funkcji zapisuje się zaczynając małą literą, każde następne słowo rozpoczyna się wielką literą (tzw. konwencja wielbłądzia), np. mojePolePrywatne. Dla pozostałych identyfikatorów każde słowo składowe rozpoczyna się dużą literą (tzw. konwencja pascalowa), np. MojaKlasa [20].

Słowa kluczowe to wyrazy zarezerwowane, tj. nie mogą stanowić identyfikatora.

Tabela 12 przedstawia przykłady takich słów.

Tabela 12. Słowa kluczowe w C# [21]

abstract	as	base	bool

break	byte	case	catch
char	checked	class	const
continue	decimal	default	delegate
do	double	else	enum
event	explicit	extern	false
finally	fixed	float	for
foreach	goto	if	implicit
in	in (generic modifier)	int	interface
internal	is	lock	long
namespace	new	null	object
operator	out	out (generic modifier)	override
params	private	protected	public
readonly	ref	return	sbyte
sealed	short	sizeof	stackalloc
static	string	struct	switch
this	throw	true	try
typeof	uint	ulong	unchecked
unsafe	ushort	using	using static
virtual	void	volatile	while

Przykłady

Listing 1 przedstawia przykładowy program „Hello, World!” napisany w C#.

Listing 1. Przykładowy kod programu "Hello, World!" napisany w języku C#

```
// Hello2.cs
using System;
public class Hello2 {
    public static void Main() {
        Console.WriteLine(“Hello, World!”);
    }
}
```

Listing 2 pokazuje, jak stworzyć i usunąć plik oraz zapisać i odczytać dane.

Listing 2. Przykład usunięcia, zapisu i odczytu pliku.

```
// sample C# code for basic file I/O operations
//exceptions ignored for code simplicity

class TestFileIO
{
   static void Main()
   {
       string fileName = "test.txt"; //a sample file name

       //delete the filr if it exists
       if (System.IO.File.Exists(fileName))
       {
           System.IO.File.Delete(fileName);
       }

       //create the file
       using (System.IO.FileStream fs = System.IO.File.Create(fileName, 102
4))
       {
            //add some information to the file
            byte[] info = new System.Text.UTF8Encoding(true).GetBytes("This
 is text in the file.");
            fs.Write(info, 0, info.Length);
       }
       //open the file and read it back
       using (System.IO.StreamReader sr = System.IO.File.OpenText(fileName)
)
       {
           string s = "";
           while ((s = sr.ReadLine()) != null)
           {
              System.Console.WriteLine(s);
            }
         }
     }
}
```

LINQ

LINQ (ang. Language Integrated Query) umożliwia odpytywanie lokalnych kolekcji i zdalnych źródeł danych.

Listing 3 przedstawia przykład metody w repository, która używa linq na wyrażeniach z lambdą.

Listing 3. LINQ na wyrażeniach z lambdą

```
      public IEnumerable<VeterinaryAppointment> GetALLFutureVeterinaryAppoi
ntments()
        {
            return Entities.Where(va => va.AppointmentDate == null)
                .Include(x => x.AppointedBy)
                .Include(x => x.Veterinary)
                .Include(x => x.HorsesToExamine);
        }
// Entities reprezentuje tabelę w bazie danych zawierającą obiekty "Veterin
aryAppointments", jest to kolekcja typu DbSet<VeterinaryAppointment>
// Na powyższej kolekcji wywoływana jest metoda LINQ "Where()" przyjmująca
jako argument warunek zwracający wartość wartość true/false
// Następnie metody "Include()" odpowiadają za złączenie tabel połączonych
z tabelą VeterinaryAppointments poprzez asocjacje
// Zapytanie SQL do serwera wykorzystuje dyrektywy SELECT, WHERE i JOIN
```

4.6 Platforma ASP.NET-MVC 5

ASP.NET MVC 5 jest środowiskiem służącym do budowy aplikacji internetowych, wykorzystującym wzorzec MVC. Obecna wersja, 5.2.3, została wydana 2 września 2015 roku [22].

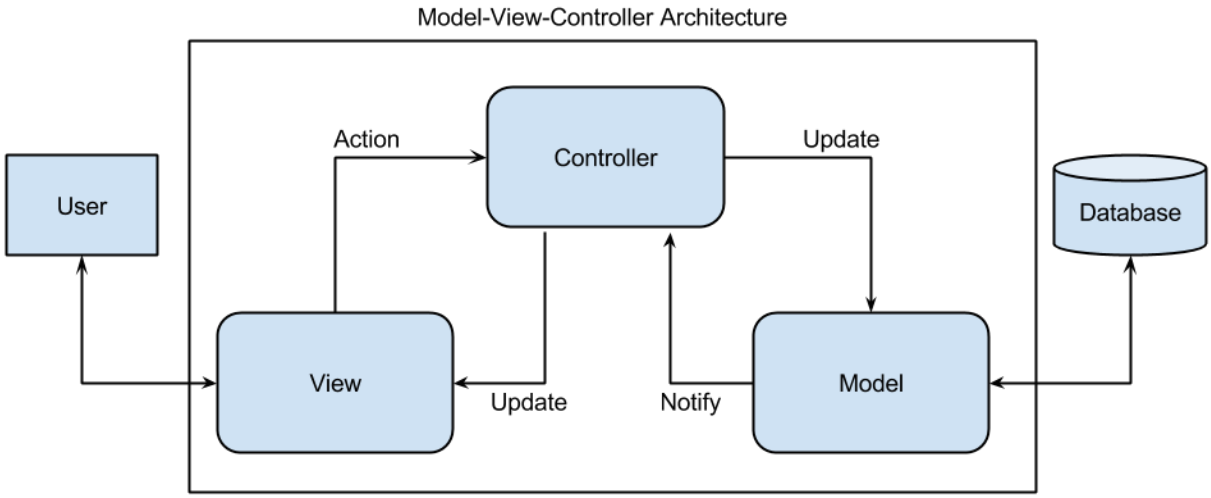
MVC jest wzorcem architektonicznym, którego pełna nazwa to Model-View-Controller. Aplikacje wykorzystujące wzorzec MVC są eleganckie w swojej prostocie, tworzone z myślą o testach oraz łatwe w utrzymaniu [23].

Wzorzec MVC składa się z trzech głównych warstw:

- Modelu, odpowiedzialnego za interakcję z bazą danych. Reprezentuje tabelę (lub kilka tabel) w bazie danych

- Widoku, który zawiera wizualne przedstawienie strony. Do pisania widoków/podczas pisania widoków wykorzystuje się HTML, CSS, JavaScript oraz Razor View Engine (pisany w języku C#, generujący odpowiedni kod HTML).
- Kontrolera, który jest „łącznikiem" między modelem i widokiem, realizuje on większość logiki biznesowej.

Rysunek 20 przedstawia architekturę MVC.



Rysunek 20. Architektura MVC

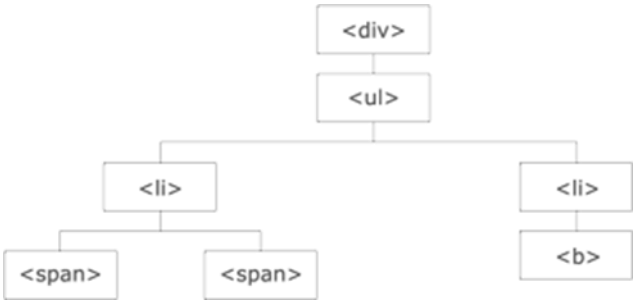
Widok odpowiedzialny jest za prezentację danych użytkownikowi. Gdy użytkownik wykona jakąś akcję następuje przekierowanie do kontrolera, skąd zostaje wysłana odpowiednia aktualizacja do modelu. W modelu następuje wywołanie lub aktualizacja bazy danych. Następnie zmiana, która zaszła z modelu, zostaje przekazana do kontrolera, skąd trafia do widoku, gdzie jest prezentowana użytkownikowi.

4.7 jQuery

jQuery jest biblioteką JavaScript stworzoną w 2006 roku. Popularność zyskała ze względu na swoją prostotę oraz duże możliwości.

Lista podstawowych funkcjonalności jQuery:

1. Manipulacja drzewem DOM. Rysunek 21 przedstawia schemat drzewa elementów HTML.



Rysunek 21. Schemat HTML jako drzewo DOM [Źródło: https://pdf.helion.pl/pjqiii/pjqiii.pdf]

2. Obsługa zdarzeń. jQuery oferuje sposób przechwytywania zdarzeń poprzez metody skrótowe m.in. kliknięcie lub najechanie na link przez użytkownika bez zmian w kodzie HTML.
3. Wsparcie przeglądarek. Współpracuje z najpopularniejszymi przeglądarkami takimi jak: FireFox 2+, Safari 3.0+, Opera 9.0+, Chrome, IE 6.0+
4. Animacje. Biblioteka posiada zbiór metod i funkcji pozwalających na kreatywne tworzenie stron internetowych, eliminuje konieczność dostosowywania kodu.
5. Obsługa Ajax. Wysyłanie żądania HTTP do serwera z poziomu strony www.

Przykłady użycia jQuery

jQuery wykorzystuje składnię łańcuchową. Każdy łańcuch rozpoczyna się znakiem **$**, a poszczególne polecenia są łączone w łańcuchy za pomocą kropki. Listing 4 ilustruje zastosowanie jQuery:

Listing 4. Przykład składni łańcuchowej. Źródło: Opracowanie własne.

```
$('pierwszy').css({color. : 'red'}).hide('slow').show(3000);

$('pierwszy')
    .css({background : 'red'})
    .hide('slow')
    .show(3000);
```

Efektem końcowym kodu przedstawionego na listingu 4 będzie przycisk przedstawiony na rysunku 22, który ukrywamy za pomocą funkcji sslw po czym pokazujemy za pomocą funkcji show.

Tabela 13 [24] prezentuje listę prostych efektów które udostępnia biblioteka jQuery.

Tabela 13. Podstawowe efekty jQuery.

Podstawowe efekty jQuery	
Nazwa efektu	Funkcja
show()	Pokazuje element.
hide()	Ukrywa element.
toggle()	Powoduje przełączenie między efektami pokazywania i ukrywania przy użyciu zdarzenia click.
slideDown()	Powoduje zsuwanie elementu w dół.
slideUp()	Powoduje przesuwanie elementu w górę.
slideToggle()	Powoduje przełączenie między efektami przesuwania elementu w górę i w dół.
fadeIn()	Powoduje zmniejszanie przezroczystości elementu.
fadeOut()	Powoduje zwiększanie przezroczystości elementu.
fadeTo()	Powoduje uzyskanie określonej nieprzezroczystości elementu.

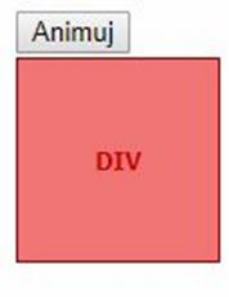
Przykład użycia Animacji w jQuery

W wielu przypadkach może zajść potrzeba animacji danych elementów na stronie. Można to zrobić za pomocą metody `animate()`, którą przedstawia listing 5.

Listing 5.Przykład animacji metoda `animate()`. Źródło:Opracowanie własne

```
$("#go").on('click', function(){
    $("#block").animate(
        {
            width: "500px",
            opacity: 0.4,
            fontSize: "3em",
            borderWidth: "10px"
          },
          1500
     );
});
```

Rysunek 22 przedstawia element przed kliknięciem przycisku Animuj.



Rysunek 22. Element przed zastosowaniem metody `animate()`.

Rysunek 23 przedstawia element po kliknięciu na przycisk animacji.

Rysunek 23. Element po zastosowaniu metody `animate()`.

4.8 HTML/CSS

System informatyczny składa się z części wewnętrznej, gdzie schowana jest cała logika aplikacji oraz części wierzchniej tzw. front-end'u, który w sposób łatwy i przyjazny pozwoli korzystać z dostępnych funkcjonalności użytkownikowi systemu. Aby SIDOS wyglądał profesjonalnie oraz był przystępny dla każdego, do projektowania interfejsu użyliśmy języków HTML oraz CSS, a także opartych na nich narzędzi Bootstrap i AdminLTE.

HTML

Historia języka HTML rozpoczęła się w roku 1991 wraz z jego pierwszą publiczną prezentacją. Dokonał tego Sir Timothy Berners-Lee, który po dziś dzień jest przewodniczącym Konsorcjum W3C, zajmującym się ustanawianiem standardów pisania i przesyłu stron WWW. Język HTML składa się z komponentów. Najważniejszym komponentem są znaczniki. Oprócz tego występują typy danych, referencje znakowe, deklaracje typu dokumentu i odwołania

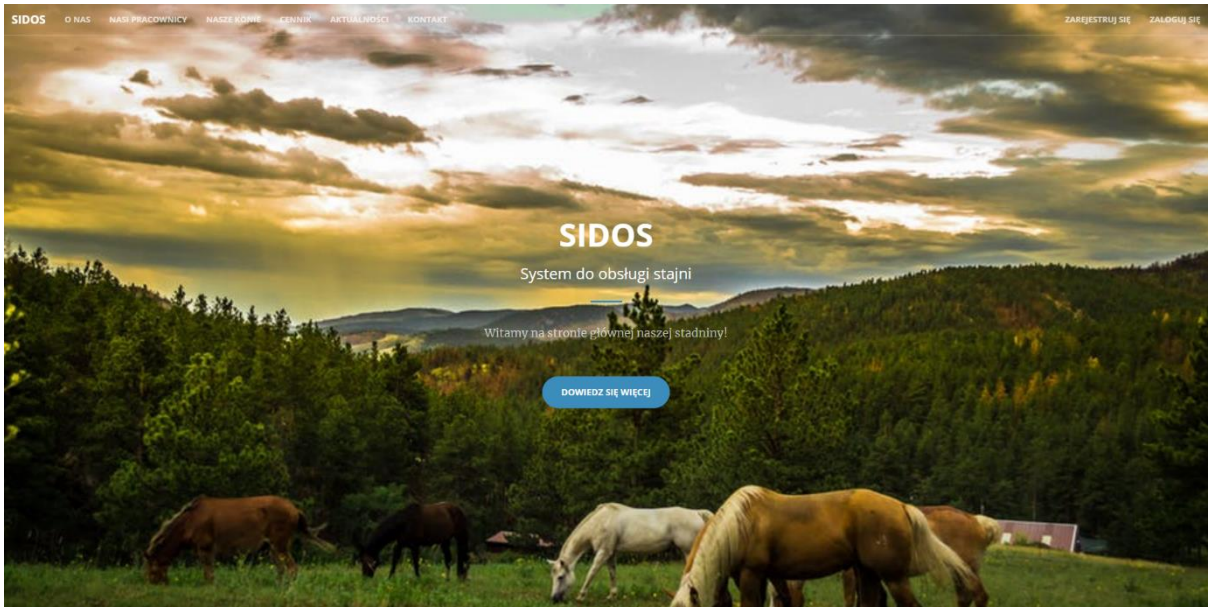
w postaci encji. W najnowszej, piątej wersji języka HTML wprowadzono wiele udogodnień, jak na przykład elementy *audio* i *video*, dzięki którym możemy odtwarzać pliki bez potrzeby instalacji dodatkowych wtyczek. Pojawiły się nowe atrybuty takie jak *email*, *URL*, *time* i *date*. Strony internetowe mogą dzięki najnowszej wersji HTML przechowywać dane lokalnie w przeglądarce użytkownika. Dzięki temu zawartość strony jest wczytywana szybciej oraz bezpieczniej, a także niepotrzebne są już *http cookies* tzw. ciasteczka.

Język HTML stanowi podstawową część warstwy wierzchniej wielu współczesnych systemów. Nie inaczej jest w przypadku systemu SIDOS, gdzie każdy widok wykorzystuje elementy tego narzędzia. Listing 6 przedstawia kod odpowiedzialny za implementację panelu nawigacyjnego strony startowej aplikacji SIDOS.

Listing 6. Kod odpowiedzialny za implementację panelu nawigacyjnego znajdującego się na samej górze strony startowej systemu SIDOS.

```
<nav id="mainNav" class="navbar navbar default navbar-fixed-top">
    <div class="container-fluid">
       <!-- Brand and toggle get grouped for better mobile display -->
       <div class="navbar-header">
           <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#bs-example-navbar-collapse-1">
           <span class="sr-only">Toggle navigantion</span> Menu <i
class="fa fa-bars"></i>
           </button> <a class="navbar-brand page-scroll" href="#page-
top">SIDOS</a>
      </div>
      <!-- COllect the nav links, forms and other content for toggling -->
      <ul class="nav navbar-nav navbar-left">
         <li>
             <a class="page-scroll" href="#about">O nas</a>
         </li>
         <li>
             <a class="page-scroll" href="#employees">Nasi pracownicy</a>
         </li>
         <li>
             <a class="page-scroll" href="#horses">Nasze konie</a>
         </li>
         <li>
             <a class="page-scroll" href="#pricelist">Cennik</a>
         </li>
         <li>
             <a class="page-scroll" href="#news">Aktualności</a>
         </li>
         <li>
             <a class="page-scroll" href="#contact">Kontakt</a>
         </li>
      </ul>
      @Html.Partial("_LoginPartial")
  </div>
</nav>
```

Rysunek 24 przedstawia stronę startową tzw. landing page. Pasek nawigacyjny znajduje się na samej górze. Po kliknięciu na którykolwiek przycisk strona przewija się do odpowiedniej sekcji. Dostęp do tej części systemu ma każdy gość odwiedzający nasz serwis. Nie jest wymagane logowanie, aby korzystać z funkcjonalności tu zawartych. Jest to wizytówka systemu, dlatego duży nacisk został położony na odpowiedni design tego elementu.



Rysunek 24 Strona startowa tzw. landing page.

CSS

CSS to język opisujący jak elementy HTML powinny wyglądać. Został zaprezentowany w 1996r. przez konsorcjum W3C, podobnie jak język HTML. Został stworzony w celu odseparowania treści dokumentu od sposobu jego prezentacji. Dzięki temu dokument jest łatwiejszy do po przeglądania, ułatwia wprowadzanie zmian. Przed powstaniem CSS wszelkie informacje dotyczące wyglądu strony były zawarte w znacznikach HTML. Dzięki jego powstaniu dane te można przenieść do osobnego pliku. Listing 7 przedstawia fragment kodu HTML oraz CSS, dzięki któremu tworzy się prosty dokument z własnym stylem paragrafów. Wynik jego przetworzenia przez przeglądarkę można zaobserwować na rysunku 25.

Listing 7 Fragment kodu HTML oraz CSS, dzięki któremu tworzy się prosty dokument z własnym stylem paragrafów.

```
<!DOCTYPE html>
<html>
   <head>
```

```
    <style>
      p {
      color: red;
      text-align: center;
      }
    </style>
  </head>
  <body>
    <p>Hello World!</p>
    <p>These paragraphs are styled with CSS.</p>
  </body>
</html>
```

Rysunek 25. Wynik przetworzenia kodu przez przeglądarkę internetową.

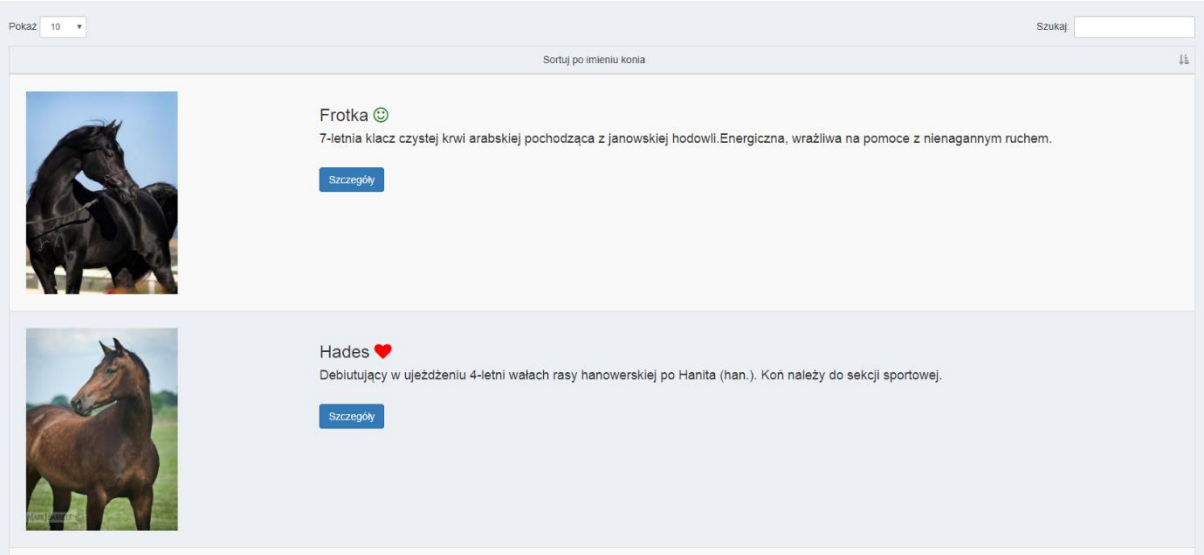
Dzięki zastosowaniu tych samych klas CSS wygląd naszego systemu w różnych miejscach jest taki sam. Tym samym tworzymy pewnego rodzaju schemat graficzny, przez który nasz system jest łatwo i szybko identyfikowalny oraz intuicyjny w obsłudze. Listing 8 przedstawia kod odpowiadający za tworzenie tabelki, która, dzięki zastosowaniu klas CSS, jest taka sama w każdej części aplikacji.

Listing 8. Tworzenie tabelki, która wielokrotnie jest wykorzystywana w systemie.

```
<div id="example_wrapper" class="dataTables_wrapper form-inline dt-
bootstrap">
    <table id="example" class="table table-striped table-bordered
dataTable" cellspacing="0" width="100%" role="grid" aria-
describedby="example_info" style="width: 100%;">
        <thread>
            <tr role="row">
                <th class="sorting text-center h5" tabindex="0" aria-
controls="example" rowspan="1" colspan="1" aria-label="Position: activate
to sort column ascending" style="width: 222px;">
                    Sortuj po imieniu konia
                </th>
            </tr>
        </thread>
        <tbody>...</tbody>
```

```
    </table>
</div>
```

Na rysunku 26 zaobserwować można efekt widoczny po stronie użytkownika systemu.



Rysunek 26. Widok użytkownika systemu.

AdminLTE

AdminLTE jest to szablon WebApp o otwartym kodzie źródłowym dla pulpitów i paneli kontrolnych administratora. Jest to responsywny szablon HTML oparty na szkielecie CSS Bootstrap 3. Wykorzystuje wszystkie elementy Bootstrap do projektowania i zmieniania stylu wielu często używanych wtyczek w celu stworzenia spójnego projektu, który może być używany jako interfejs użytkownika dla aplikacji zaplecza.

AdminLTE opiera się na modułowej konstrukcji, która pozwala na łatwe dostosowanie strony na potrzeby danego projektu. Podstawą AdminLTE są dwa podstawowe frameworki.

- Bootstrap 3
- jQuery 1.11+

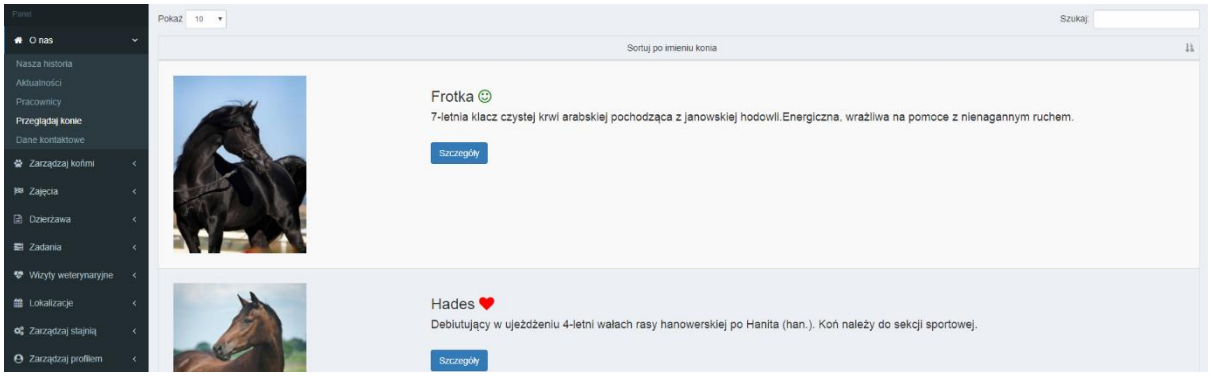
Schematu AdminLTE używamy do głównych kontrolek nawigacyjnych, umożliwiających przemieszczanie się pomiędzy poszczególnymi funkcjonalnościami w systemie. W przypadku naszego projektu wykorzystujemy głównie współdzielony layout. Menu nawigacyjne znajdujące się u góry i z lewej strony ekranu pozostaje zawsze takie samo, niezależnie od podstrony, którą odwiedzamy. Dodatkowo, widok, w którym się znajdujemy,

jest podświetlony na biało jako aktywny, a cała sekcja tematyczna jest rozwinięta. Design zawarty w tej bibliotece bardzo dobrze łączy się z pozostałą częścią front-endu systemu. Kolor niebieski wyłonił się jako kolor przewodni aplikacji SIDOS. Kod pokazany na listingu 9 jest odpowiedzialny za wyświetlanie paska bocznego tzw. sidebar menu. Zawarte są w nim odnośniki, które przenoszą nas w różne części naszego systemu. Części te są podzielone tematycznie, tak, aby odnalezienie poszczególnych funkcjonalności było jak najprostsze.

Listing 9. Kawałek kodu odpowiedzialny za implementację menu nawigacyjnego.

```
<!-- Sidebar Menu -->
<ul class="sidebar-menu">
  <li class="header">Panel</li>
  <!-- Optionally, you can add icons to the links -->
  <li class="treeview">
   <a href="#">
   <i class=fa fa-home"</i>
   <span>O nas</span>
   <span class="pull-right-container">
   <i class="fa fa-angle-left pull-right"></i>
   </span>
   </a>
   <ul class="treeview-menu">
    <li>@Html.ActionLink("Nasza historia", "StableHistory", "StableDetails")</li>
    <li>@Html.ActionLink("Aktualności", "Index", "News")</li>
    <li>@Html.ActionLink("Pracownicy", "Index", "Employee")</li>
    <li>@Html.ActionLink("Przeglądaj konie", "Index", "Horse")</li>
    <li>@Html.ActionLink("Dane kontaktowe", "ContactDetails", "StableDetails")</li>
   </ul>
  </li>
 <li class="treeview"> . . . </li>
 <li class="treeview"> . . . </li>
 <li class="treeview"> . . . </li>
 <li class="treeview"> . . . </li>
 <li class="treeview"> . . . </li>
 <li class="treeview"> . . . </li>
 <li class="treeview"> . . . </li>
 <li class="treeview"> . . . </li>
</ul>
<!-- /.sidebar-menu -->
```

Na rysunku 27 widać wszystkie dostępne sekcje w naszym systemie. W tym konkretnym przypadku użytkownik znajdował się w widoku „Przeglądaj konie”, który znajduje się w zakładce „O nas”.



Rysunek 27. Aktywna zakładka O nas w pasku nawigacyjnym po lewej stronie ekranu.

Bootstrap

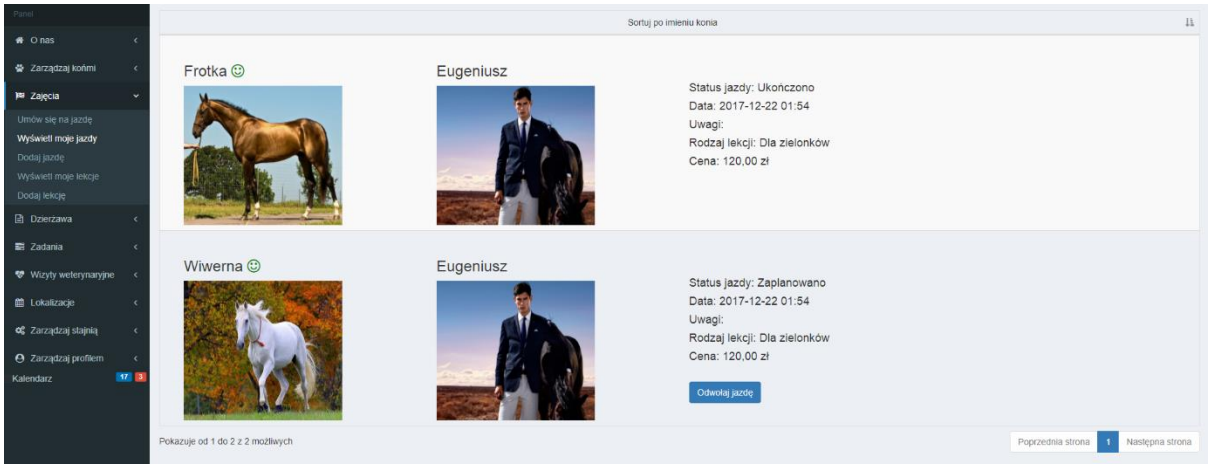
Bootstrap powstał w 2010 roku. Opracowali go Mark Otto oraz Jacob Thornton pracując w firmie Twitter. Jest to projekt typu open-source, który jest w tej chwili drugim najliczniej obserwowanym projektem na platformie GitHub. Bootstrap składa się z szablonów opartych na językach HTML, CSS jak również może zawierać opcjonalne rozszerzenia JavaScript. Stosowany jest tylko i wyłącznie do front-endu, czyli do poprawy wyglądu strony. Przed powstaniem tego framework'u programiści byli zmuszeni do korzystania z wielu różnych bibliotek w celu uzyskania przejrzystego i estetycznego designu strony. Wiązało się to z licznymi problemami integracyjnymi oraz zwiększało koszty utrzymania projektu. Najnowszą, stabilną wersją jest v3.3.7, jednak trwają już pracę nad wersją 4, która jest aktualnie w fazie beta. Zastosowanie Bootstrap'a w projekcie daje szereg korzyści. Oprócz poprawienia wyglądu strony, możemy w bardzo łatwy sposób skorzystać z dostępnych komponentów pozwalających na dostosowanie wyświetlanych elementów do własnych potrzeb. Warto nadmienić, że korzystanie z tego framework'a powoduje, że nasza strona domyślnie staje się responsywna, a jej wygląd dopasowuje się do rozdzielczości urządzenia, na którym jest wyświetlana. Dzięki bibliotece Bootstrap nasz system jest responsywny, a poszczególne kontrolki mają nowoczesny wygląd oraz spójny design z resztą systemu. Opisywany wcześniej AdminLTE także w dużej części korzysta z tej platformy programistycznej. Bardzo przydatnym sposobem prezentowania danych jest tzw. grid system. Pozwala ono na tworzenie maksymalnie dwunastu, dowolnej szerokości kolumn w widoku, które same się dostosowują do rozdzielczości ekranu, na którym są wyświetlane. Na listingu 10 pokazany jest kawałek kodu odpowiedzialny za implementację tabelki wyświetlającej jazdy, na które umówił się użytkownik. Widok został podzielony na 12 kolumn, z czego

połowę zajmują zdjęcia konia i użytkownika, a pozostałą część szczegółowe informacje o jeździe.

Listing 10. Podział widoku na 12 części.

```
<tr role="row" class="odd">
 <td class="sorting_1">
  <div class="content">
   <div class="col-md-3">
    <p class="text-bold h3">@item.Horse.Name <a href="#" data-toggle="tooltip"
title="@item.Horse.Character.GetDisplayName()"><i class="@CharacterType" aria-
hidden="true" style="@CharacterStyle"></i> </a></p>
    <img src="@Url.Content(item.Horse.MainPicture)" height="220" width="250" />
   </div>
   <div class="col-md-3">
    <p class="text-bold h3">@item.Lesson.Instructor.FirstName</p>
    <img src="@Url.Content(item.Lesson.Instructor.ProfilePicture)" height="220"
width="250" />
   </div>
   <div class="col-md-6">
    <br />
    <br />
    <p class="h4">@Html.DisplayNameFor(model => model.Status):
@item.Status.GetDisplayName()</p> @if (item.Status ==
SIDOS.Core.Enums.RideStatus.Cancelled) { <p class="h4">@Html.DisplayNameFor(model
=> model.CancellationCause): @Html.DisplayFor(model =>item.CancellationCause)</p> }
    <p class="h4">@Html.DisplayNameFor(model => model.Lesson.Date):
@item.Lesson.Date.ToShortDateString() @item.Lesson.Date.ToShortTimeString()</p> <p
class="h4">@Html.DisplayNameFor(model => model.Comments): @Html.DisplayFor(model =>
item.Comments)</p>
    <p class="h4">@Html.DisplayNameFor(model => model.Lesson.LessonType):
@item.Lesson.LessonType.Name</p>
    <p class="h4">@Html.DisplayNameFor(model => model.Lesson.Price):
@Html.DisplayFor(model => item.Lesson.Price) zł </p>
    <br />
    @if(User.IsInRole("Instructor") == false && item.Status ==
SIDOS.Core.Enums.RideStatus.Planned)
    {
     @Html.ActionLink("Odwołaj jazdę", "CancelMyRide", new { id = item.Id }, new {
@class = "btn btn-primary" })
    }
     @if(User.IsInRole("Instructor") == true)
    {
     @Html.ActionLink("Edytuj jazdę", "EditMyRide", new { id = item.Id }, new {
@class = "btn btn-primary })
     }
   </div>
  </div>
 </td>
</tr>
```

Rysunek 28 przedstawia spis jazd, na które zapisany jest użytkownik. Dzięki zastosowaniu biblioteki Bootstrap strona po zmniejszeniu okna dopasuje się do rozmiaru i odpowiednio zmieni układ wyświetlania danych. Korzystanie z systemu SIDOS na urządzeniach mobilnych nie powinno więc być żadnym problemem.



Rysunek 28. Efekt końcowy zastosowania podziału strony na kolumny.

4.9 AJAX

Skrót AJAX (ang. Asynchronous JavaScript And XML) oznacza: asynchroniczny JavaScript i XML. Korzystanie ze strony wykorzystującej AJAX odbywa się bez konieczności przeładowywania jej w całości. Elementy zawierające dane po stronie klienta są zmieniane asynchronicznie, natomiast pozostałe składowe strony np. master page i komponenty strony nie posiadające żadnych danych do pobrania z serwera pozostają bez zmian. Technologie JavaScript i XMLHttpRequest są odpowiedzialne za wymianę danych z serwerem, dzięki czemu nie ma potrzeby ponownego przeładowywania całej strony [25].

Do pełnego funkcjonowania AJAX potrzebuje:

- HTML (lub XHTML) i CSS do wyświetlania danych na stronie;
- DOM (Document Object Model) do dynamicznego wyświetlania danych i interakcji z nimi;
- JSON lub XML do wymiany danych i XSLT do manipulacji nimi;
- XMLHttpRequest do asynchronicznej komunikacji;
- JavaScript do połączenia wszystkich powyższych technologii [26].

Ogólnie o technice AJAX

W latach 90. XX wieku strony internetowe były głównie oparte na samych tagach HTML. Każda akcja na stronie wykonana przez użytkownika wymagała przeładowania całej treści. Było to bardzo niewygodne dla użytkowników - zawartość zmieniała się co chwila, co było uciążliwe dla osoby obsługującej dane na stronie.

W 1996 roku został opublikowany tag `iframe` dla przeglądarki Internet Explorer. Ten element strony jako pierwszy pozwalał na asynchroniczne pobieranie danych do strony. W 1998 zespół Outlook Web App stworzył koncepcję obiektu XMLHttpRequest, który potem jako XMLHTTP pojawił się w drugiej wersji biblioteki MSXML. Internet Explorer 5.0 zaczął wspierać technologię XMLHTTP w marcu 1999 roku. Funkcjonalność XMLHTTP została później zaimplementowana w pozostałych przeglądarkach internetowych, m.in. w Mozilla Firefox, Safari i Operze.

Termin AJAX pojawił się dopiero w lutym 2005 roku w artykule pt. „Ajax: A New Approach to Web Applications". Organizacja World Wide Web Consortium (W3C), tworząc pierwsze zręby standardu stron internetowych, opisała w kwietniu 2006 roku specyfikację dla XMLHttpRequest. Najnowszy wpis dotyczący tego obiektu pojawił się w styczniu 2014 roku [27].

Użycie techniki AJAX w systemie SIDOS

Technologia AJAX pozwala na pobieranie danych z serwera bez przeładowania strony. Zostało to wykorzystane do zaimplementowania przycisków zwalniających i przywracających pracowników. Dzięki temu po użyciu przycisku „Zwolnij", komórka odpowiadająca za identyfikację daty zwolnienia w tabeli odpowiednio się zaktualizuje. Będzie pokazywać datę zwolnienia. Listing 11 pokazuje kod, który jest wykorzystywany do tej akcji. Listing 12 przedstawia kod wykorzystujący technologię AJAX do przywracania pracowników. Poprzez `UpdateTargetId` Ajax odpowiednio zmodyfikuje dany element w tabeli.

Listing 11. Kod wykorzystujący technologię AJAX w przycisku zwalniania pracownika.

```
@Ajax.ActionLink("Zwolnij", "Fire", new { id = item.Id }, new AjaxOptions
{
  HttpMethod = "GET",
  UpdateTargetId = @item.Id.ToString() + "Date",
  InsertionMode = InsertionMode.Replace
}, new { @class = "btn btn-primary btn-block btn-
danger", onClick = "changeFire(this.id)", id = @item.Id.ToString() + "Fire"
 })
```

Listing 12. Kod wykorzystujący technologię AJAX w przycisku przywracania pracownika.

```
@Ajax.ActionLink("Przywróć", "ReHire", new {id = item.Id}, new AjaxOptions
{
  HttpMethod = "GET",
  UpdateTargetId = @item.Id.ToString() + "Date",
  InsertionMode = InsertionMode.Replace
}, new { @class = "btn btn-primary btn-block btn-
success", onClick = "changeReHire(this.id)", id = @item.Id.ToString() + "Re
Hire" })
```

4.10 JavaScript

Przed rozpoczęciem fazy implementacji grupa Inżynierii Oprogramowania i Baz Danych musiała się zdecydować co do technologii, które będą używane w trakcie budowania aplikacji po stronie klienta. Na rynku języków do programowania dynamicznych elementów strony dostępne są m.in. CoffeScript, TypeScript, Babel i JavaScript [28].

Jednym z ważniejszych powodów wybrania języka JavaScript była duża ilość wsparcia w sieci na różnych forach i poradnikach [29], wbudowane ułatwienia dla tego języka w IDE Visual Studio i znajomość tego języka wśród członków grupy implementacyjnej.

JavaScript (oryginalnie LiveScript albo Mocha) jest to język programowania stworzony przez firmę Netscape założoną w 1994 roku [30]. Sam język powstał rok później, a za jego twórcę podaje się Brendana Eicha, amerykańskiego hakera i programistę. W 1997 roku organizacja ECMA (European Association for Standardizing Information and Communication Systems) z siedzibą w Genewie rozpoczęła pracę nad specyfikacją języka JavaScript pozbawioną odniesień na temat interpretującego kod środowiska. Wynikiem prac została specyfikacja języków skryptowych ECMAScript, której jedną z implementacji jest JavaScript. Obecnie JavaScript jest w wersji 1.8 od dziewięciu lat, lecz Netscape we współpracy z Fundacją Mozilla pracuje nad wersją 2.0.

JavaScript – cechy

JavaScript jest słabo typowanym językiem skryptowym. Oznacza to, że nie ma ściśle określonych typów, w związku z czym wartości obiektów są dynamiczne. Jakikolwiek obiekt może mieć wartość np. ‘a’, 127 lub True. Wartości te można zmieniać, usuwać i dodawać.

Jest to również język bez żadnych klas. Istnieją rozwiązania mające je symulować, jednak w przypadku konieczności korzystania z klas z powodzeniem można sięgnąć po rozwiązania na rynku, takie jak na przykład TypeScript [31], które od początku tworzone są z myślą o nich.

Nie istnieje podział na poziomy dostępności znane z języków obiektowych takich jak np. Java, C# czy C++. Przez co nie jesteśmy w stanie określić jawnie czy dana zmienna lub metoda jest publiczna (ang. `public`) czy też prywatna (ang. `private`), bo zwyczajnie taki mechanizm nie istnieje.

JavaScript wspólnie z takimi językami jak HTML czy CSS stanowią najważniejszą część programistyczną front-endu. Są to języki często rekomendowane jako podstawa aplikacji webowych na świecie, którą musi znać każdy twórca stron internetowych [32].

JavaScript – przykłady

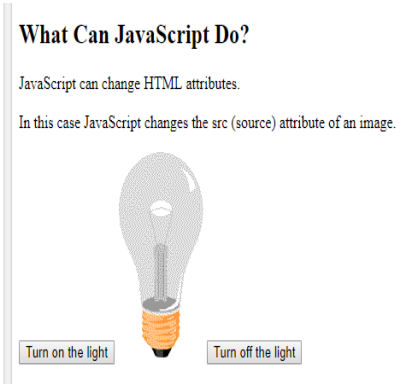
Listing 13 przedstawia JavaScript osadzony w kodzie HTML. Tworzy przycisk, po którego wciśnięciu zmienia się rozmiar elementu HTML ‘p’, czyli tekstu „JavaScript can change the style of an HTML element” na 35px.

Listing 13. Przykładowe przedstawienie osadzenia kodu JavaScript (linijka 6) w kodzie HTML

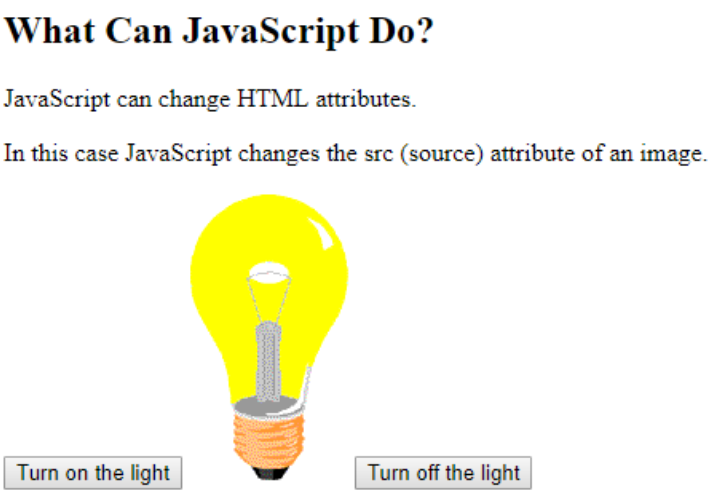
```
<!DOCTYPE>
<html>
  <body>
    <h2>What can JavaScript do?</h2>
    <p id="demo">JavaScript can change the style of an HTML element.</p>
    <button type="button"
onclick="document.getElementById('demo').style.fontSize='35px'">Click Me!</button>
   </body>
</html>
```

JavaScript, oprócz zmiany zawartości elementów CSS dla poszczególnych tagów HTML, potrafi zmieniać również wartości atrybutów samego tagu HTML np. źródło obrazka (ang. `src`), co pokazuje rysunek 29a. Po wciśnięciu przycisku „Turn on the light” źródło obrazka się zmieni i widoczna będzie zapalona żarówka, co przedstawia rysunek 29b.

```
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p>JavaScript can change HTML attributes.</p>

<p>In this case JavaScript changes the src (source) attribute of an image.</p>

<button onclick="document.getElementById('myImage').src='pic_bulbon.gif'">Turn on the light</button>

<img id="myImage" src="pic_bulboff.gif" style="width:100px">

<button onclick="document.getElementById('myImage').src='pic_bulboff.gif'">Turn off the light</button>

</body>
</html>
```



Rysunek 29a. Wygląd rysunku po zmianie właściwości src przy pomocy języka JavaScript



Rysunek 29b. Zmieniony obrazek po kliknięciu przycisku "Turn on the light".

JavaScript – podsumowanie

JavaScript jest to język skryptowy, który wymieniany jest jako jeden z 3 najważniejszych do nauczenia się przy tworzeniu frontend'u stron internetowych. Jego zaletą jest spora ilość bibliotek, pluginów i narzędzi ułatwiających jego pisanie. Jeśli chcemy, to możemy skorzystać z wielu różnych framework'ów napisanych by ułatwić programowanie w języku JavaScript m.in. AngularJS, ReactJS, Ember.js czy Meteor.js [33]. Dzięki wymienionym wcześniej udogodnieniom zdecydowaliśmy się m.in. na korzystanie z języka JavaScript przy tworzeniu systemu SIDOS.

4.11 EntityFramework

Entity Framework jest to framework, który stosuje się do mapowania obiektowo-relacyjnego (ang. Object-Relational Mapping). Wykorzystuje się go dla technologii ADO.NET [34]. Entity Framework może być wykorzystany na trzy sposoby: Model First, Code First i Database First. Model First umożliwia tworzenie w wizualnym kreatorze diagramu. Na podstawie jego generujemy język definicji danych (ang. Data Definition Language), który tworzy bazę danych. Ponadto możemy też utworzyć przy pomocy podejścia Model First modele obiektowe. Code First oznacza, że po napisaniu obiektów, generujemy z nich relacyjną bazę danych. W przypadku podejścia Database First tworzymy wcześniej bazę, a następnie przy jej pomocy realizujemy część obiektową. Podobnie jak w przypadku Code First może być ona automatycznie wygenerowana. Zaletą korzystania z tego frameworku jest minimalizacja

tzw. niezgodności impedancji[3] oraz oddzielenie warstwy związanej z dostępem danych od aplikacji. Inną zaletą jest to, że LINQ to Entities pozwala nam na wykorzystywanie języka LINQ do operacji na bazie. Możemy dzięki temu pracować tylko i wyłącznie z danymi w środowisku obiektowym, bez potrzeby pisania zapytań w języku SQL, bo są one generowane na podstawie języka LINQ.

Mapowanie przy pomocy Entity Framework przebiega w następujący sposób:

a.) Pola klasy są mapowane na atrybuty w bazie danych
b.) Kolekcje *ICollection* są realizacjami relacji typu „wiele"
c.) Referencje do obiektu danej klasy są realizacjami relacji typu „jeden"
d.) Klasy w modelu obiektowym są przekształcane na klasy w modelu relacyjnym

Do budowy systemu SIDOS skorzystaliśmy z podejścia Code First. W SIDOS.Core w folderze Modele znajdują się klasy i ich pola, które zostały ustalone w trakcie przebiegu fazy analizy i projektowania. Przykładowa klasa `Task` będzie mapowana w następujący sposób:

a.) Sama klasa `Task` będzie miała swoją tabelę w bazie o nazwie `Task`
b.) Pola typu `Name` i `Description` zostaną zmienione na atrybuty typu nvarchar
c.) Pola typu `Type` i `Status` będą miały wartość int, która identyfikuje dany typ enum
d.) Referencje o nazwach `AssignedTo` i `AssignedBy`. Jedna z nich będzie kluczem obcym dla tabeli `Employee`, a druga dla tabeli `AssignedBy`. Uniqueidentifier jest ich typem danych
e.) Pole `Dosage` zostanie zamienione na typ float

Rysunek 30 w interfejsie graficznym w Visual Studio pokazuje już gotową bazę Na tym samym rysunku poniżej możemy zobaczyć instrukcję DDL, która tworzy tabelę na podstawie danych w modelu.

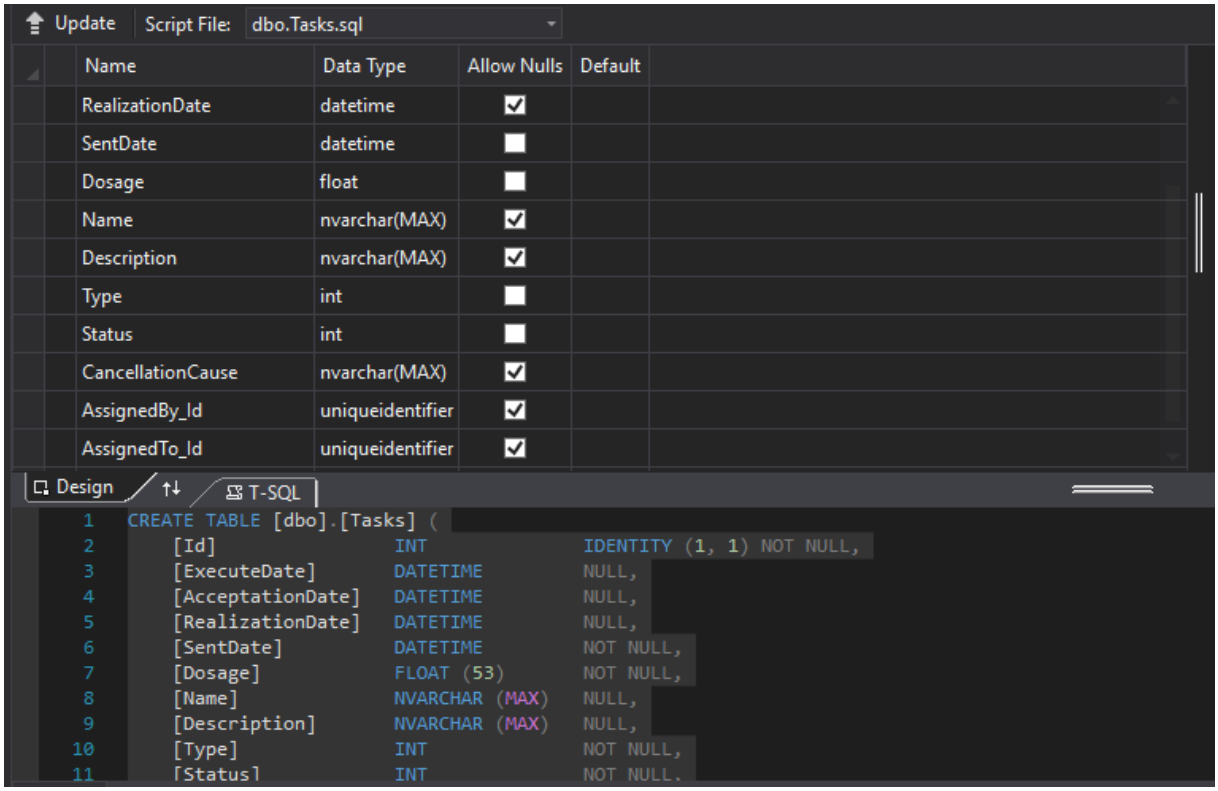
Należy jeszcze zwrócić uwagę na to jak są realizowane różne ograniczenia dla atrybutów w bazie np. NOT NULL, MAX lub MIN. Jednym ze sposobów ustalania ograniczeń jest dodawanie *DataAnnotations* dla pól w modelu. Następnie w trakcie mapowania danego modelu brane są pod uwagę adnotacje. Listing 14 pokazuje adnotację, która utworzy

[3] Niezgodność impedancji występuje na styku dwóch różnych modeli, np. obiektowego oraz relacyjnego. Więcej informacji można znaleźć w [35].

ograniczenie NOT NULL na kolumnie `SentDate`, która w modelu obiektowym jest polem typu DateTime.

Listing 14. Przykład użycia Data Annotations. Pole SentDate będzie posiadało ograniczenie NOT NULL w bazie danych

```
   public class Task : BaseModel
{
   ///Realna data kiedy Pracownik wykonał zadanie
   public DateTime? ExecuteDate ( get; set; }

   ///Data zaakceptowania zadania przez Pracownika
   [Display(Name = "Data akceptacji")]
   public DateTime? AcceptationDate { get; set; }

   ///Data kiedy pracownik ma wykonać zadanie
   [Display(Name = "Data wykonania")]
   public DateTime RealizationDate { get; set; }

   ///Data kiedy wysłano zadanie do Pracownika
   [Required]
   [Display(Name = "Data wysłania")]
   public DateTime SentDate { get; set; }
```



Rysunek 30. Widok Server Explorer w Visual Studio, gdzie widać zmapowane dane dla modelu Task

5 System do zarządzania ośrodkiem jeździeckim

W rozdziale tym omówione zostaną założenia architektoniczne systemu, interfejs użytkowników, komunikacja z poziomu aplikacji oraz w jaki sposób system będzie wspierał pracowników stadniny.

5.1 Założenia architektoniczne

Wprowadzenie

Projekt ze względu na swoją skalę został podzielony na kilka części w celu uporządkowania go i oddzielenia elementów, które znacząco różnią się od siebie w systemie. Jeśli chodzi o część użytkową, to wydzielono dwie części: strona startowa i kokpit. Strona powitalna jest dostępna dla każdej osoby i przedstawia ogólne informacje o stajni. Natomiast kokpit przechowuje część dostępną głównie dla pracowników stajni i osób wynajmujących lub dzierżawiących konie. Z tego miejsca też zarządza się stajnią.

Ze względu na utrzymanie kodu, projekt podzielono na takie części jak: SIDOS.Core, SIDOS.DataAccessLayer, SIDOS.Tests i SIDOS.Web.

W utrzymaniu projektu miały nam też pomóc wzorce projektowe, m.in. wstrzykiwanie zależności, wzorzec repozytorium i model-widok-kontroler.

Architektura projektu

Podział ze względu na część użytkową:

a.) Strona powitalna. Część dostępna zarówno dla osób zalogowanych jak i niezalogowanych. Ma głównie charakter informacyjny, przedstawiający najważniejsze informacje o ośrodku jeździeckim.
b.) Kokpit. Część przeznaczona wyłącznie dla użytkowników zalogowanych. Mogą być to m.in. osób chcących się zapisać na jazdy, pracowników i kierowników. Służy też do zarządzania systemem np. do dodawania, usuwania, edycji obiektów.

Podział ze względu na utrzymanie kodu:

a.) SIDOS.Core. Składowa przechowująca logikę biznesową. Znajdują się tutaj elementy takie jak Modele, które są mapowane przez Entity Framework do bazy danych i interfejsy potrzebne do m.in. realizacji wzorca Repository Pattern.

b.) SIDOS.DataAccessLayer. Tutaj przechowywana jest populacja bazy danych, zaimplementowane interfejsy z wyżej wymienionej części SIDOS.Core jak i `SidosDbContext` przechowujący obiekty typu DbSet potrzebne do zarządzania bazą.
c.) SIDOS.Tests. Część stworzona do przeprowadzania testów.
d.) SIDOS.Web. Część odpowiedzialna za implementację logiki biznesowej. Znajdują się tutaj wszystkie elementy front-end'u, Kontrolery, ViewModele, skrypty CSS, skrypty JavaScript, Serwisy jak i Dependency Injection.

Wstrzykiwanie zależności

Wstrzykiwanie zależności (ang. Dependency Injection) używane jest do zmienienia połączeń pomiędzy obiektami klas w taki sposób, aby usunąć bezpośrednie zależności pomiędzy nimi. Realizacja tego wzorca polega na przekazaniu poprzez parametr konstruktora lub metodę, gotowej, automatycznie utworzonej wcześniej instancji obiektu [36]. W systemie SIDOS dokonywane jest wstrzyknięcie zależności poprzez konstruktor.

Wzorzec ten wykorzystuje się ze względu na uniezależnienie klas wysokiego poziomu od tych niższego poziomu. Operować mamy w tym wzorcu na abstrakcyjnej implementacji, a nie konkretnej danego modułu. Dzięki temu tworzy się łatwiejsze do testowania obiekty. Ponadto późniejsza zmiana implementacji naszych komponentów powinna być prostsza, bo nie mamy mocnych powiązań pomiędzy klasami wyższego poziomu od tych niższego. W systemie SIDOS wykorzystuje się wstrzyknięcie zależności dla klasy `UnitOfWork`.

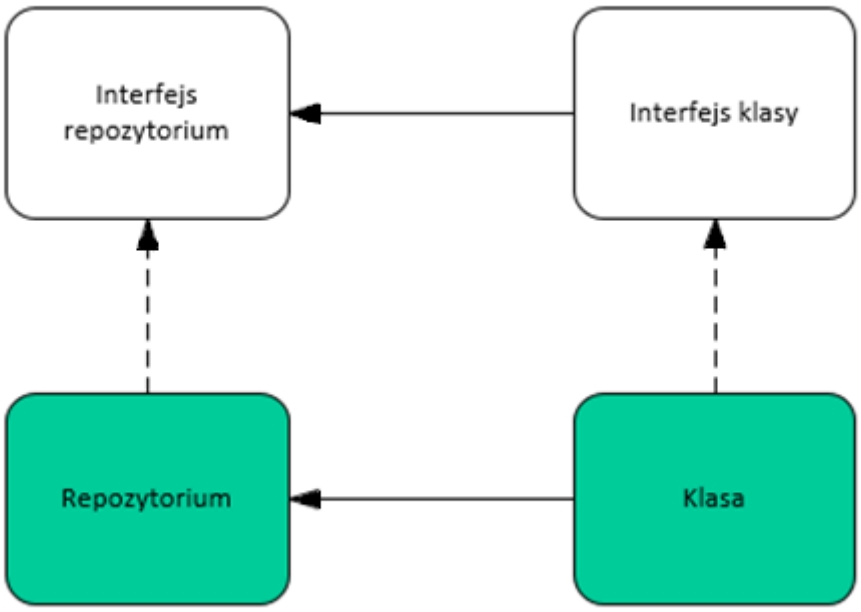
Wzorzec repozytorium

Wzorzec projektowy, który polega na utworzeniu warstwy pośredniczącej pomiędzy operacjami na logice biznesowej, a przechowywanymi danymi [37]. Wzorzec implementuje się poprzez repozytoria (ang. Repository) i jednostki pracy (ang. Unit of Work). Repozytorium powinno przechowywać kilka podstawowych metod, które mogą być wykorzystywane przez wszystkie nasze klasy biznesowe w projekcie np. dodaj, usuń, zwróć dany obiekt, zwróć wszystkie obiekty, znajdź poprzez wskazany predykat. Jednostka pracy z kolei ma posiadać wszystkie obiekty typu `DbSet`, a ponadto metodę odpowiedzialną za aktualizacje stanu bazy danych.

Wzorzec wykorzystuje się, aby wyeliminować powtarzające się zapytania do bazy danych, by uniezależnić projekt od wykorzystywanych frameworków i ułatwić testowanie systemu.

Nową warstwę tworzy się w naszym projekcie poprzez wykonanie repozytorium, interfejsu repozytorium, interfejsów dla poszczególnych klas biznesowych i jednostki pracy (ang. Unit of Work). Każdy interfejs klasy musi dziedziczyć po interfejsie repozytorium. Z kolei zaimplementowana już klasa ma dziedziczyć swój odpowiedni interfejs, a ponadto repozytorium. Samo repozytorium natomiast dziedziczy po swoim interfejsie. Rysunek 31 przedstawia sposób działania hierarchii dla wzorca repozytorium. Jednostka pracy przechowuje obiekty danych klas biznesowych i jeden obiekt reprezentujący kontekst naszej bazy danych.

Repository Pattern został wykorzystany w projekcie głównie w celu uniknięcia powtarzających się zapytań. Dodatkowo zapewnić ma oddzielenie Entity Framework od naszej logiki biznesowej. Dzięki temu teoretycznie w przyszłości będzie można zmienić Framework, jeśli zajdzie taka potrzeba.



Rysunek 31. Diagram przedstawiający dziedziczenie dla wzorca repozytorium

5.2 Interfejs użytkowników

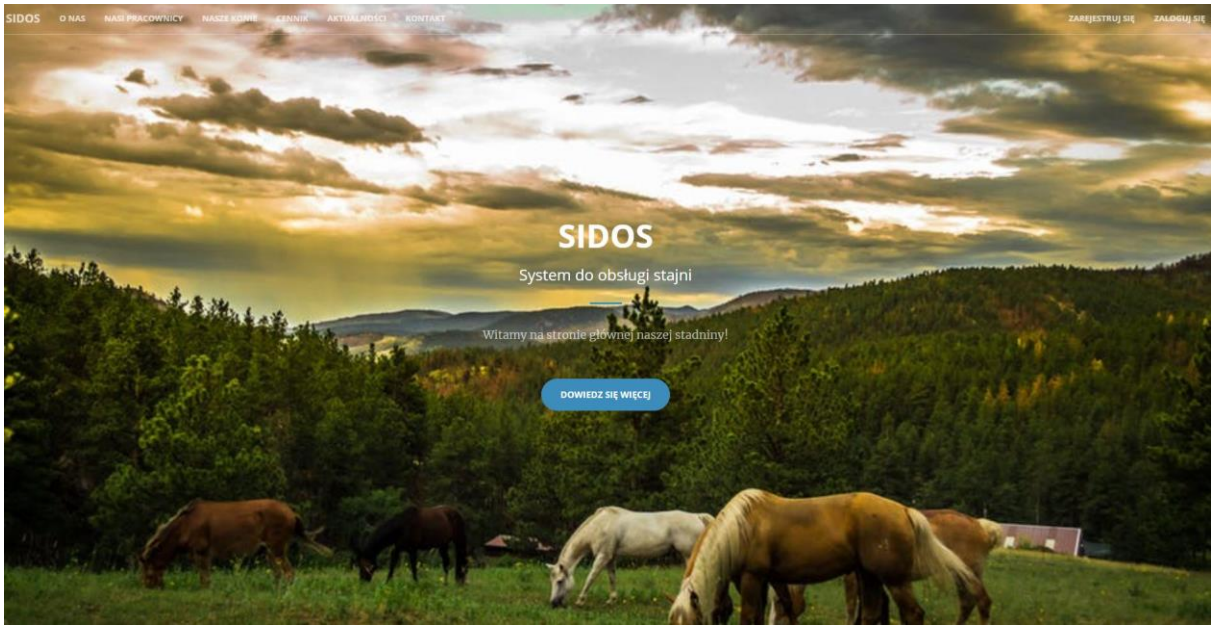
Interfejs użytkownika to jeden z najważniejszych elementów oprogramowania, decydujący o jakości i funkcjonalności strony internetowej. W dzisiejszych czasach poprzez popularyzacje oraz dostępność wielu stron internetowych o podobnej tematyce istnieje szereg różnych rozwiązań interfejsów graficznych. Użytkownik wybierze rozwiązanie najbardziej intuicyjne i przejrzyste, które nie będzie sprawiało kłopotów w nawigacji. Jednocześnie przy projektowaniu nie możemy zapomnieć o kolorystyce i skalowalności oraz responsywności systemu.

SIDOS jest nowoczesnym systemem bazującym na minimalistycznym wyglądzie zawierającym proste schematy, co zapewnia bezproblemową nawigację po stronie klienta. Głównym motywem wykorzystanym na stronie jest panel AdminLTE, opisany w rozdziale 4.8, który łączy ze sobą walory estetyczne z użytecznością.

Pierwszym widokiem, który napotyka użytkownik systemu, jest ogólnodostępna strona powitalna (*Landing Page*), przedstawiająca najważniejsze informacje o ośrodku, takie jak:

- Krótki opis stajni (Rysunek 33)
- Pracownicy (Rysunek 34)
- Konie (Rysunek 35)
- Cennik (Rysunek 36)
- Aktualności (Rysunek 37)
- Kontakt (Rysunek 38)

Rysunek 32 przedstawia stronę powitalną.



Rysunek 32. Strona powitalna. Źródło: Opracowanie własne

Przycisk „Dowiedz się więcej" znajdujący się w centrum przenosi użytkownika do widoku „O stajni", w którym przedstawiona jest krótka charakterystyka stajni oraz istotne informacje dodatkowe. Treść artykułu została skrócona, aby poprawić czytelność na stronie. Rysunek 33 prezentuje widok O stajni.

Rysunek 33. Widok O stajni. Źródło: Opracowanie Własne

Powtarzalność schematów znacznie poprawia nawigację na stronie, potencjalny użytkownik dokonując interakcji z danym widokiem będzie w stanie przewidzieć, gdzie zostanie przeniesiony. Zastosowanie przycisku „Poznaj Nas" działa analogicznie do przycisku „Dowiedz się więcej". Przycisk przenosi odwiedzającego na część strony powitalnej poświęconą zespołowi pracowników pracujących w stajni. Rysunek 34 przedstawia widok „Nasi pracownicy".

Rysunek 34. Widok Nasi pracownicy. Źródło: Opracowanie własne

Jednym z ważniejszych czynników decydujących o tym, czy klient umówi się na jazdę konną, są instruktorzy oraz kadra pracująca w danym miejscu. Prezentacja została zrealizowana w postaci przesuwających się kafelków, na których przedstawiono zdjęcia poszczególnych pracowników. Widok dodatkowo umożliwia interakcję użytkownika z interfejsem. Po najechaniu kursorem na danego pracownika wyświetlany jest krótki, treściwy opis.

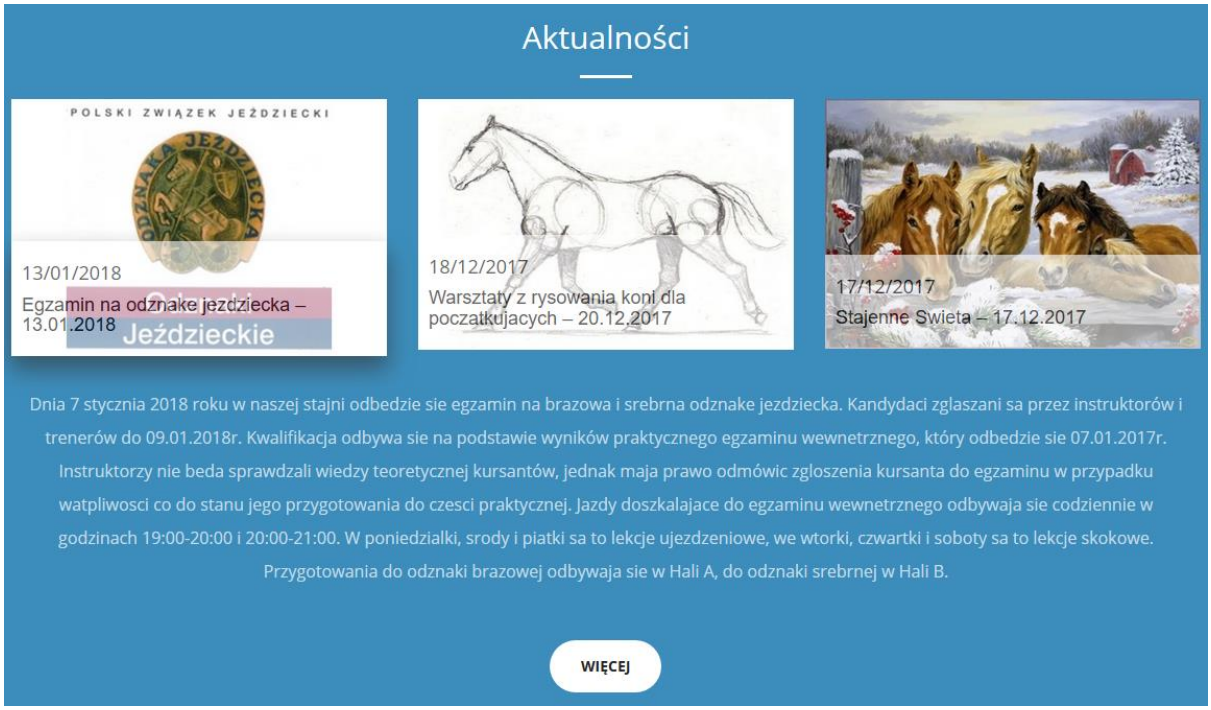
Analogicznie został zrealizowany widok „Nasze konie", w którym, również pod postacią kafelków, wyświetlana jest lista koni. Podobnie jak w przypadku pracowników, istnieje funkcja szybkiego podglądu cech danego konia wyświetlonego na liście. Rysunek 35 przedstawia widok „Nasze Konie".

Rysunek 35. Widok Nasze konie. Źródło: Opracowanie własne.

Rysunek 36 przedstawia Cennik, który zapewnia nam szybki dostęp do zapoznania się z ofertą stadniny. Wyróżniono 3 warianty lekcji, są one statycznie wpisane do bazy danych i wyświetlane na stronie.

Rysunek 36.Widok Cennik. Źródło: Opracowanie własne.

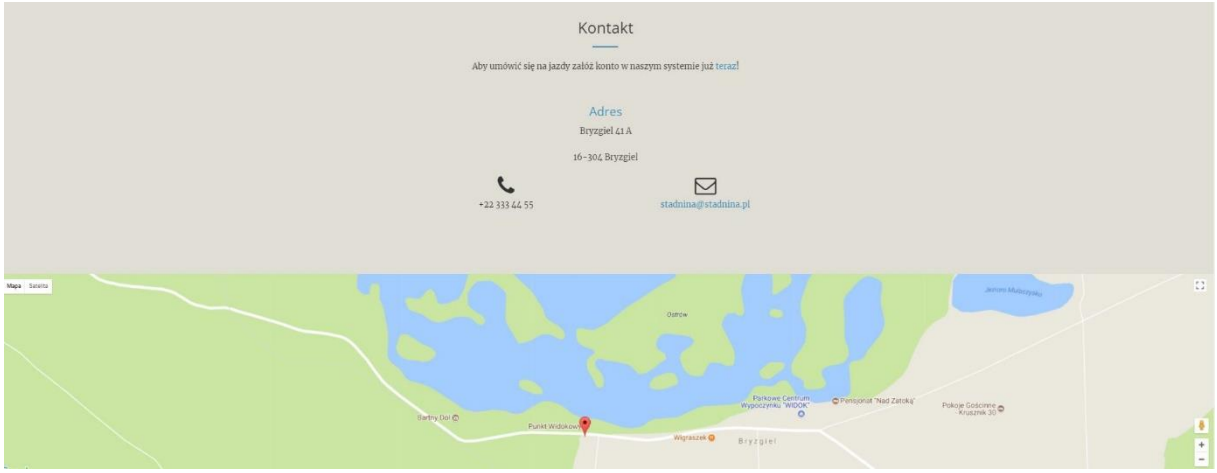
W dobie szybkiego dostępu do informacji funkcjonalność aktualności jest podstawą do przyciągnięcia potencjalnych klientów stadniny. Jest to widok, w którym zamieszczone są informacje o kursach, wydarzeniach i innych atrakcjach oferowanych przez stadninę. Wyświetlane są one w postaci notki zawierającej treść, datę oraz opis wydarzenia. Elementem charakterystycznym dla każdego ogłoszenia jest zdjęcie, które ma na celu zainteresować użytkownika. Przycisk „Więcej" działa wyłącznie dla zalogowanego użytkownika i pozwala mu na podgląd innych aktualności. Rysunek 37 przedstawia widok Aktualności.



Rysunek 37. Widok Aktualności. Źródło: Opracowanie własne.

Elementem znajdującym się na końcu strony jest zakładka Kontakt. Rysunek 38 przedstawia mapę wyświetlaną za pomocą Aplikacji Google Map oraz podstawowe dane kontaktowe wyrażone w przejrzystym układzie.

Przycisk „tutaj” przekierowuje użytkownika do strony logowania. Zalogowanie do systemu pozwoli mu na dokładniejsze zapoznanie się z innymi możliwościami oferowanymi przez SIDOS[4].



Rysunek 38. Widok Kontakt. Źródło: Opracowanie własne

[4] SIDOS – System Informatyczny Do Obsługi Stajni

Nawigacja

Nawigacja na stronie docelowej systemu została podzielona na 6 głównych zakładek oraz przyciski logowania i rejestracji znajdujących się w przypiętym do strony menu. Rysunek 39 przedstawia widok menu.

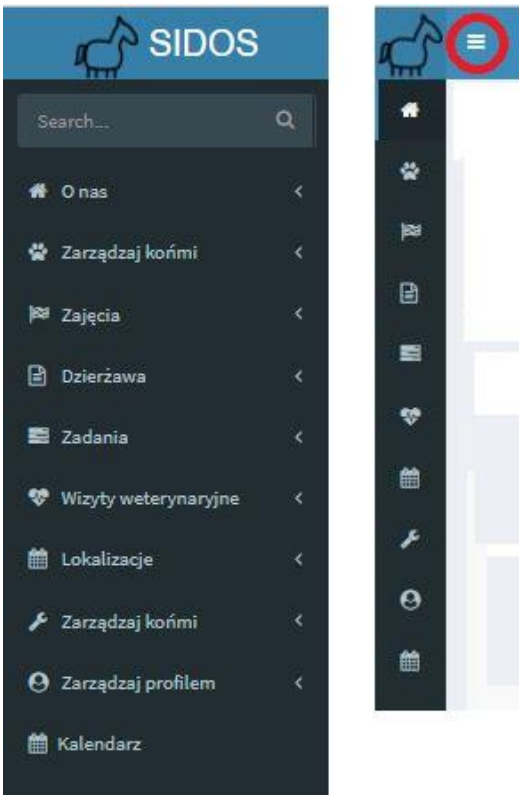
Rysunek 39. Menu. Źródło: Opracowanie własne.

Zakładki wchodzące w skład menu:

- O nas (Rysunek 33)
- Nasi pracownicy (Rysunek 34)
- Nasze konie (Rysunek 35)
- Cennik (Rysunek 36)
- Aktualności (Rysunek 37)
- Kontakt (Rysunek 38)
- Zarejestruj się (Rysunek 39)
- Zaloguj się (Rysunek 39)

Nawigacja jest fundamentem dobrze zaprojektowanej strony internetowej. W większości przypadków sprawdzone modele są najlepsze – jednym z nich jest klasyczne menu w górnej części strony. Ma to ogromny wpływ na wyszukiwanie treści na stronie oraz swobodne poruszanie się między widokami. System SIDOS skierowany jest do grupy odbiorców w zróżnicowanym przedziale wiekowym, dzięki czemu osoby starsze czy mające mniejsze doświadczenie w obsłudze stron internetowych powinny z łatwością odnaleźć się na stronie.

Nawigacja w systemie z poziomu zalogowanego użytkownika realizowana jest poprzez znajdujący się po lewej stronie „sidebar" zawierający zakładki odnoszące się do funkcji systemu oraz wyszukiwarkę przedstawiony na rysunku 40. W celu przejścia do swobodnego przeglądania w trybie pełnoekranowym strony użytkownik ma możliwość zwinięcia prawego paska bocznego poprzez przycisk zaznaczony kolorem czerwonym widoczny po prawej stronie rysunku 40.



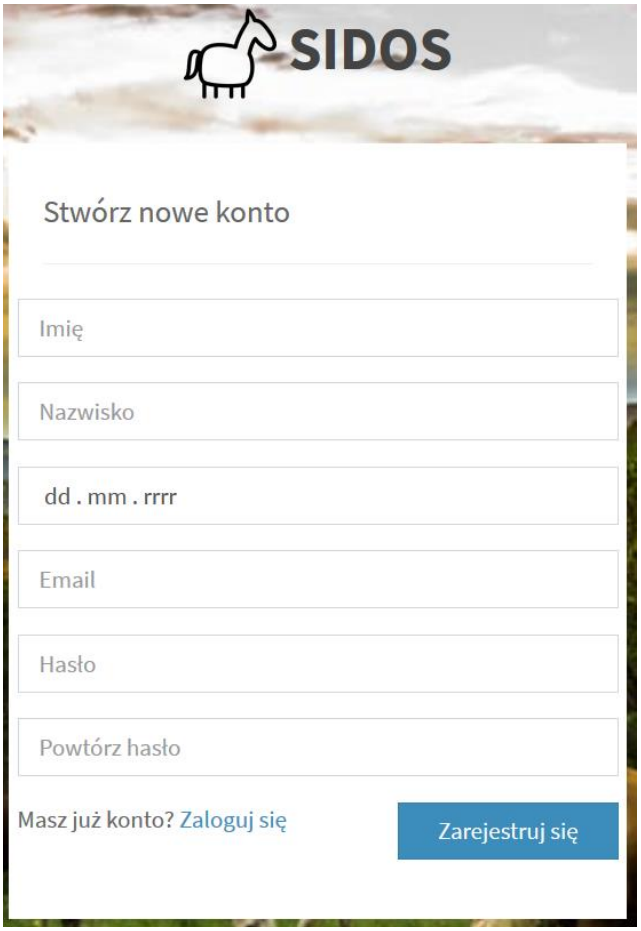
Rysunek 40.Menu użytkownika zalogowanego rozsunięte oraz schowane. Źródło: Opracowanie własne.

Stałym elementem nawigacyjnym zalogowanego użytkownika systemu jest również nagłówek znajdujący się w górnym prawym rogu przedstawiony na rysunku 41. Zawiera dodatkowe notyfikacje związanie z wiadomościami oraz informacje o koncie, na które

jesteśmy zalogowani poprzez wyświetlenie właściwego adresu mail użytkownika i przycisk pełniący funkcje wylogowania się z systemu.

Rysunek 41. Informacje dotyczące wiadomości i konta, na które zalogowany jest użytkownik. Źródło: Opracowanie własne

Aby móc w pełni skorzystać z funkcjonalności jakie posiada system wymagane zalogowanie się użytkownika do systemu. Jeśli użytkownik nie posiada konta w systemie może je założyć poprzez wypełnienie formularza rejestracyjnego widocznego na rysunku 42.



Rysunek 42.Widok rejestracji. Źródło: Opracowanie własne.

W systemie SIDOS wyodrębniono kilka rodzajów kont:

- Kierownik
- Weterynarz

- Instruktor
- Kowal
- Stajenny
- Klient

Każde konto posiada zróżnicowany dostęp do funkcjonalności systemu w zależności do roli jaka dana osoba pełni w systemie zgodnie z diagramami przypadków użycia (xx) sporządzonymi na etapie projektowania i analizy. Konta pracowników; kowal, stajenny, instruktor, weterynarz są dodawane przez kierownika systemu pełni o n funkcje pewnego rodzaju administratora.

Rysunek 43 przedstawia widok logowania.

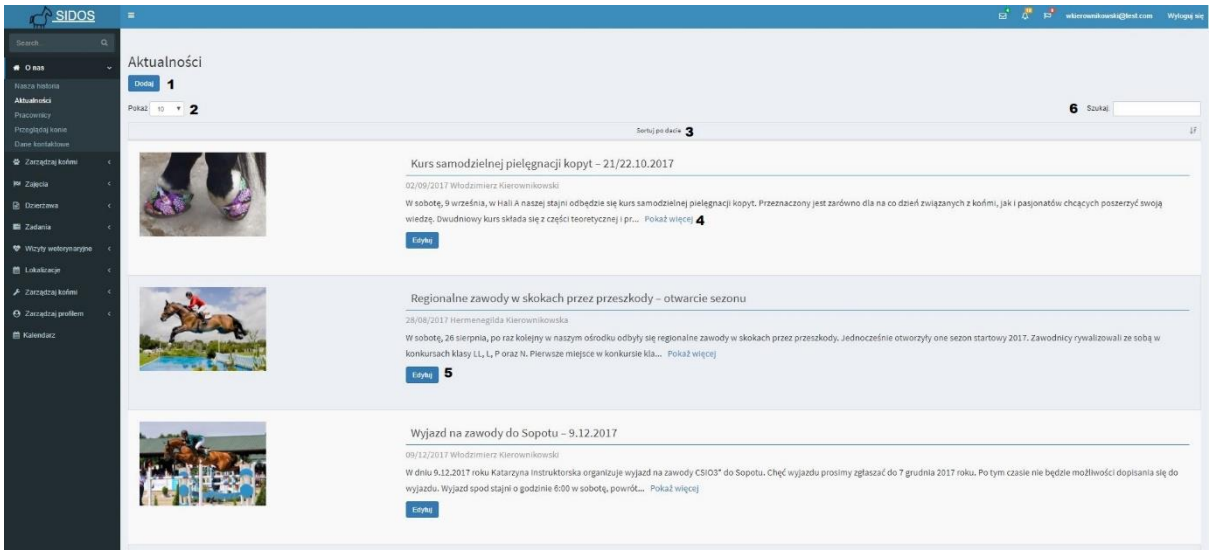
Rysunek 43.Widok logowania. Źródło: Opracowanie własne.

W celu przedstawienie wszystkich wymagań funkcjonalnych oferowanych przez system będziemy korzystali z wielu kont.

Aby w pełni przedstawić funkcjonalności dotyczące administracji systemem będziemy korzystać z konta kierownika.

Aktualności

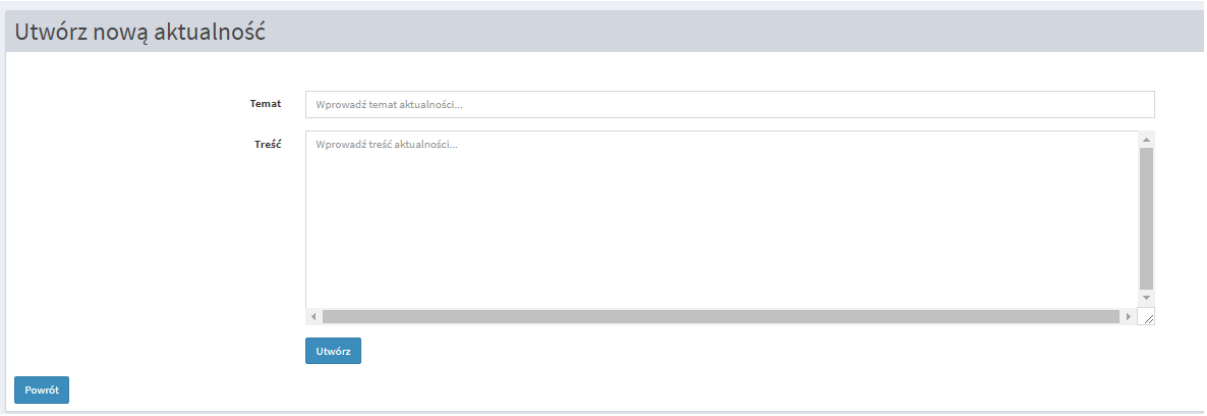
Jedną z ważniejszych funkcjonalności, którą posiada tylko kierownik jest dodawanie bieżących oraz edycja aktualności związanych z życiem stadniny. Rysunek 44 przedstawia widok aktualności z poziomu kierownika.



Rysunek 44.Widok Aktualności dla kierownika. Źródło: Opracowanie własne.

Na potrzeby widoku zaimplementowano sortowanie postów po dacie (pkt 3 – rysunek 44) oraz licznik obrazów wyświetlanych na stronie (pkt 2 – rysunek 44) co znacznie ułatwiło przeglądanie treści. Z pomocą w odnalezieniu konkretnej wartości idzie funkcja wyszukiwania (pkt 6 – rysunek 44).

Dodawanie nowej aktualności odbywa się poprzez przycisk „Dodaj", który przekierowywane nas na widok znajdujący się na rysunku 45.



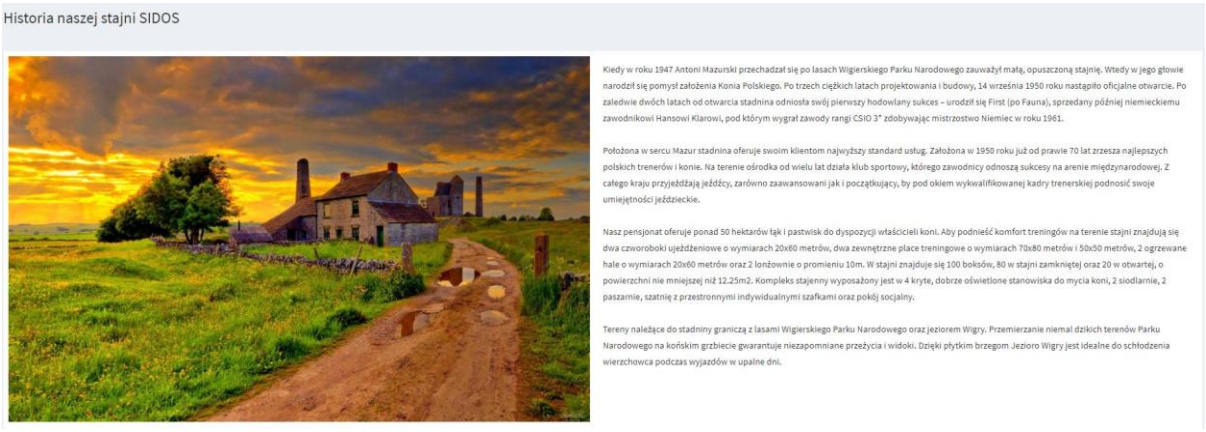
Rysunek 45. Dodaj nową aktualność. Źródło: Opracowanie własne.

Nowa aktualność zawiera obowiązkowe pola temat, treść. Nowo utworzony post domyślnie wyświetlany jest wg. daty dodania w widoku na rysunku 44 oraz jego treść jest skompresowana w celu przejrzystości przeglądania. Przycisk „pokaż więcej" (pkt 4 – rysunek 44) rozwija nam treść aktualność.

W systemie kierownik posiada możliwość edycji aktualności poprzez przycisk edycji (pkt 5 – rysunek 44) znajdujący się pod konkretną aktualnością. Poprawioną wersje możemy zapisać poprzez przycisk „Zapisz". Widok edycji został przedstawiony na rysunku 46.

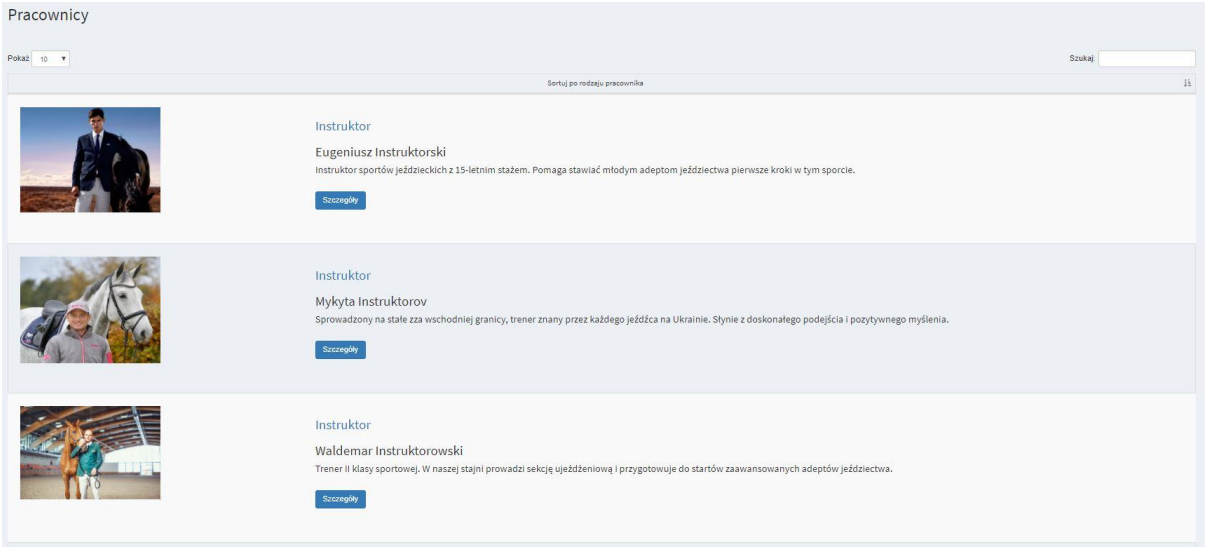
Rysunek 46.Edytuj Aktualność. Źródło: Opracowanie własne.

Zakładka nasza historia przedstawia rozszerzony opis założenia i umiejscowienia stadniny widoczny tylko dla zalogowanych użytkowników. Rysunek 47 prezentuje widok, na którym przedstawiony jest opis wraz z zdjęciem.



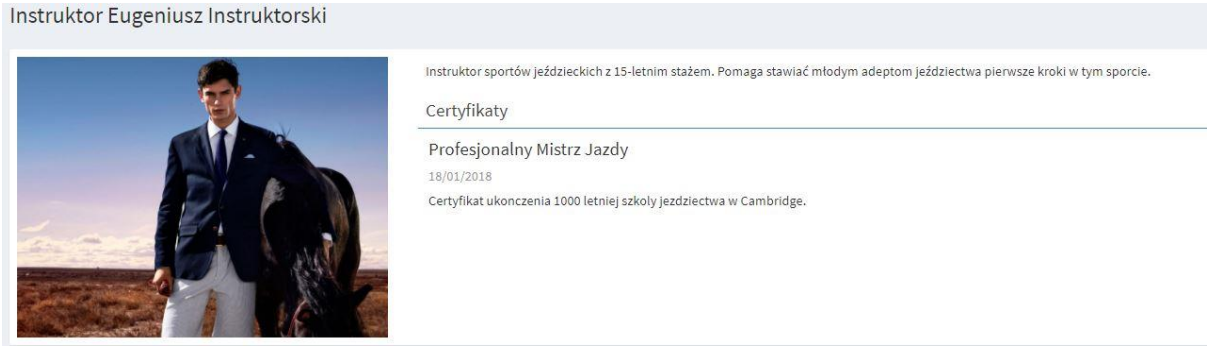
Rysunek 47. Nasza historia. Źródło: Opracowanie własne.

Widok znajdujący się na rysunku 48, do którego mają dostęp zalogowani użytkownicy systemu przedstawia profile pracowników stadniny. Na potrzeby widoku zaimplementowano funkcje sortowania alfabetycznie po nazwie stanowiska pracownika.



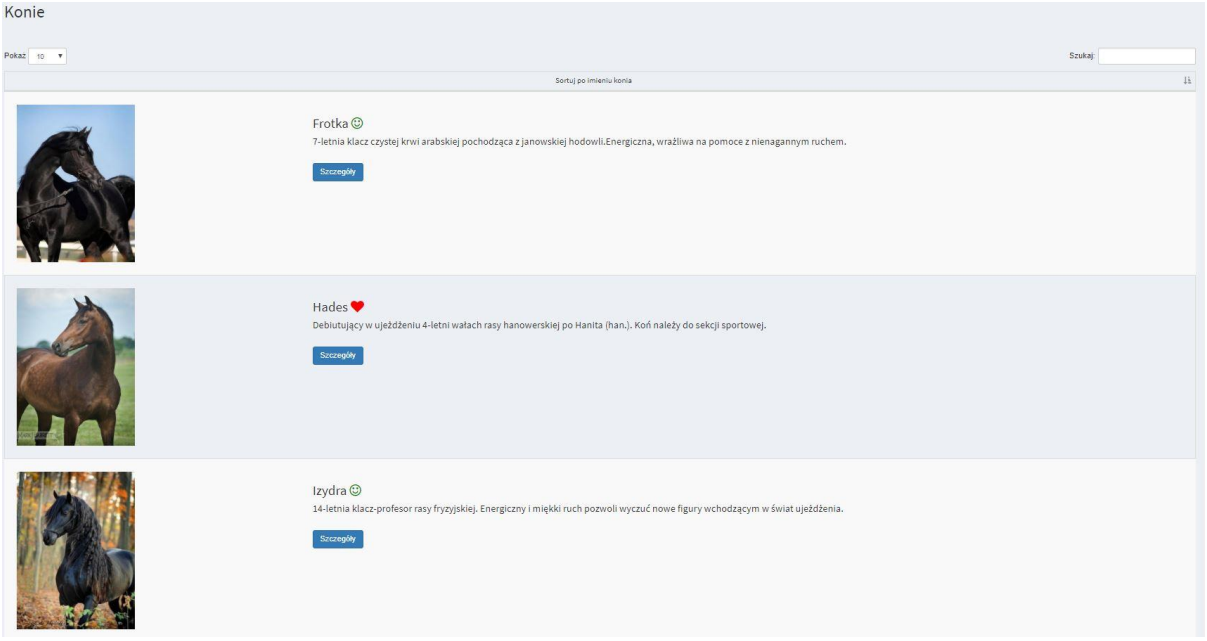
Rysunek 48.Przeglądaj listę pracowników. Źródło: Opracowanie własne.

Aby dowiedziesz się więcej o specjalizacji danego pracownika wystarczy przejść w widok szczegółowy pracownika, gdzie widnieją jego certyfikaty oraz krótki opis. Rysunek 49 przedstawia widok szczegółowy pracownika.



Rysunek 49. Widok szczegółowy pracownika. Źródło Opracowanie własne.

Widok przedstawiaj koni znajdujących się w stadninie dostępny jest dla każdego zalogowanego użytkownika. Została ona przedstawiona na rysunku 50.

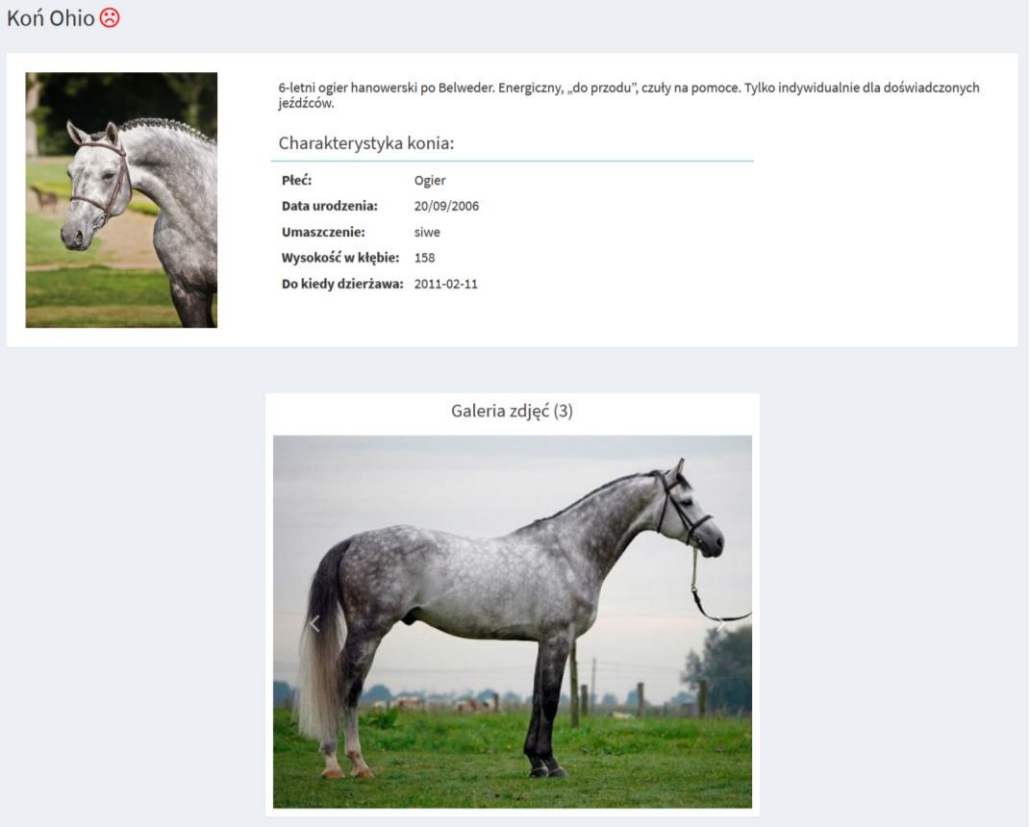


Rysunek 50. Przeglądaj listę koni. Źródło: Opracowanie własne.

Aby użytkownik mógł trafniej dopasować konia do własnych preferencji w systemie dla każdego konia wyodrębniono rodzaje charakteru. Ikona przedstawia rodzaj charakteru:

- Łagodny
- Ospały
- Narowisty
- Przyjazny

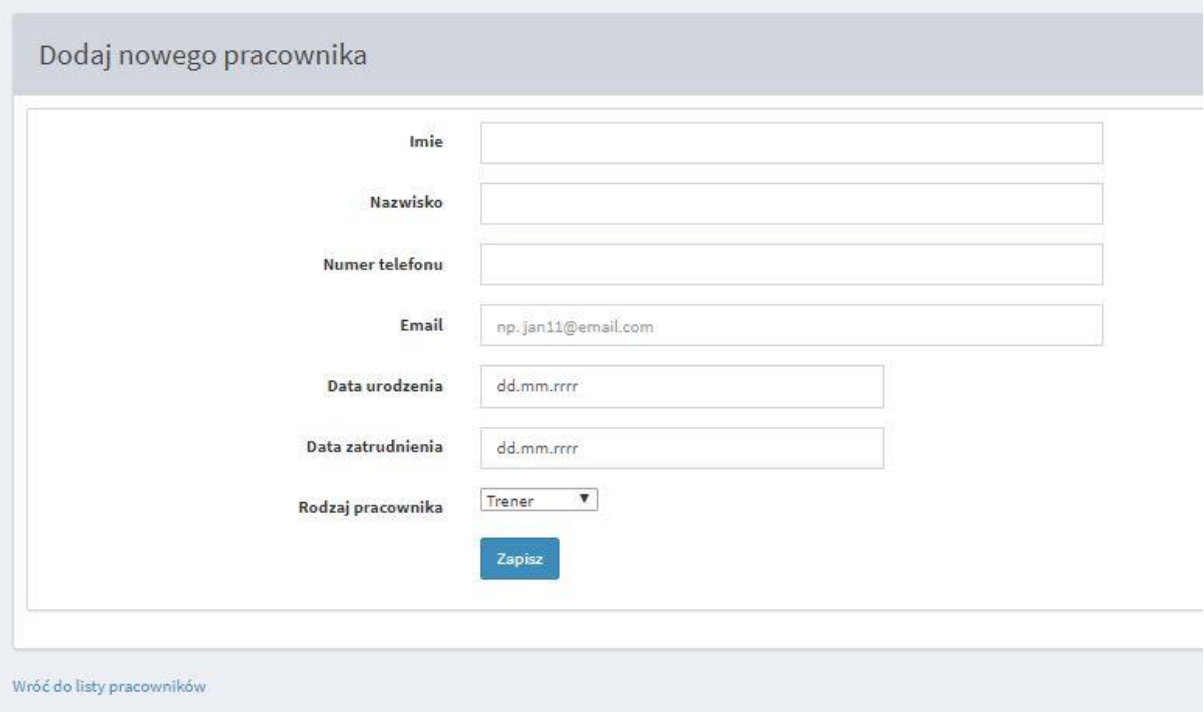
W widoku szczegółowym konia widoczne są dane konia, galeria zdjęć, opis oraz jego charakter. Rysunek 51 przedstawia szczegółowy widok profilu konia.



Rysunek 51. Widok profilu konia

Zarządzaj Stajnia

Moduł zarządzaj stajnią jest dedykowany dla kierownika. W tym miejscu ma on dostęp do funkcji dodania pracownika wraz z przypisaniem stanowiska od osoby. Widok dodania pracownika przedstawiony jest na rysunku 52.

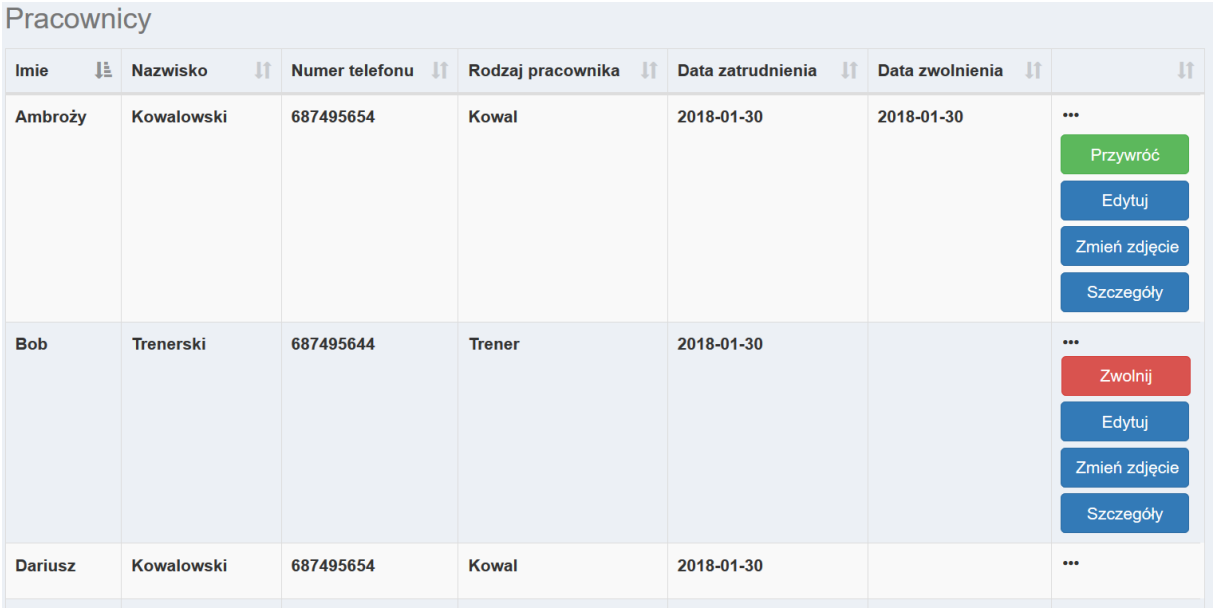


Rysunek 52.Dodaj nowego pracownika. Źródło: Opracowanie własne.

Widok reprezentujących listę zatrudnionych pracowników zawierający historie zatrudnienia danego pracownika wraz z możliwością sortowania danych po polach:

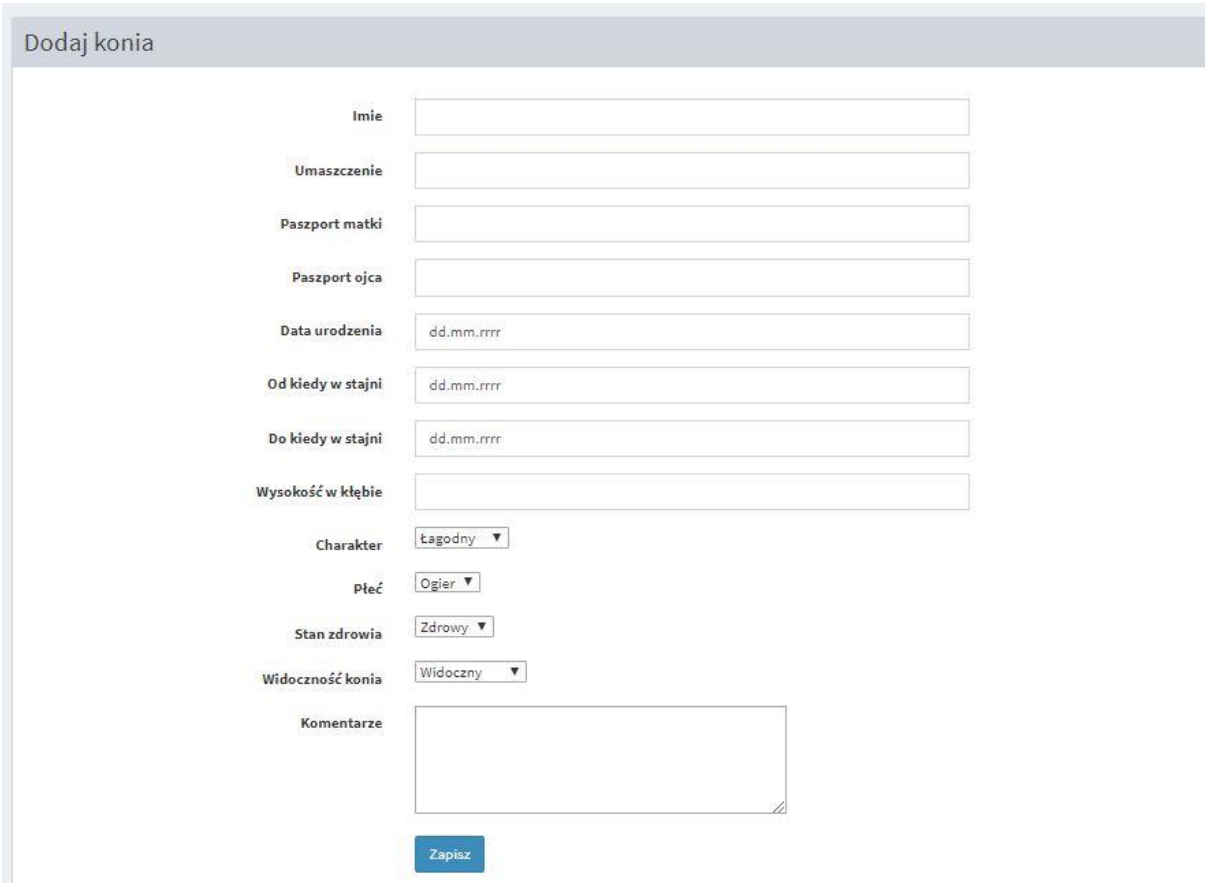
- imię
- nazwisko
- numer telefonu
- rodzaj pracownika
- data zatrudnienia
- data zwolnienia

Poprzez przycisk rozwijający funkcjonalności (trzy kropki w ostatniej kolumnie tabeli przedstawionej na rysunku 53) kierownik może zwolnić pracownika oraz przywrócić go do pracy. Dodatkowo istnieje możliwość edycji danych pracownika, zmiana zdjęcia profilowego oraz widok szczegółowy przedstawiający dane osobowe pracownika oraz jego rodzaj.



Rysunek 53.Widok wszystkich pracowników. Źródło: Opracowanie własne.

W panelu zarządzaj stajnia również odbywa się dodawanie koni do systemu. Wartości konieczne przy dodawaniu nowego konia przedstawia formularz znajdujący się na rysunku 54.



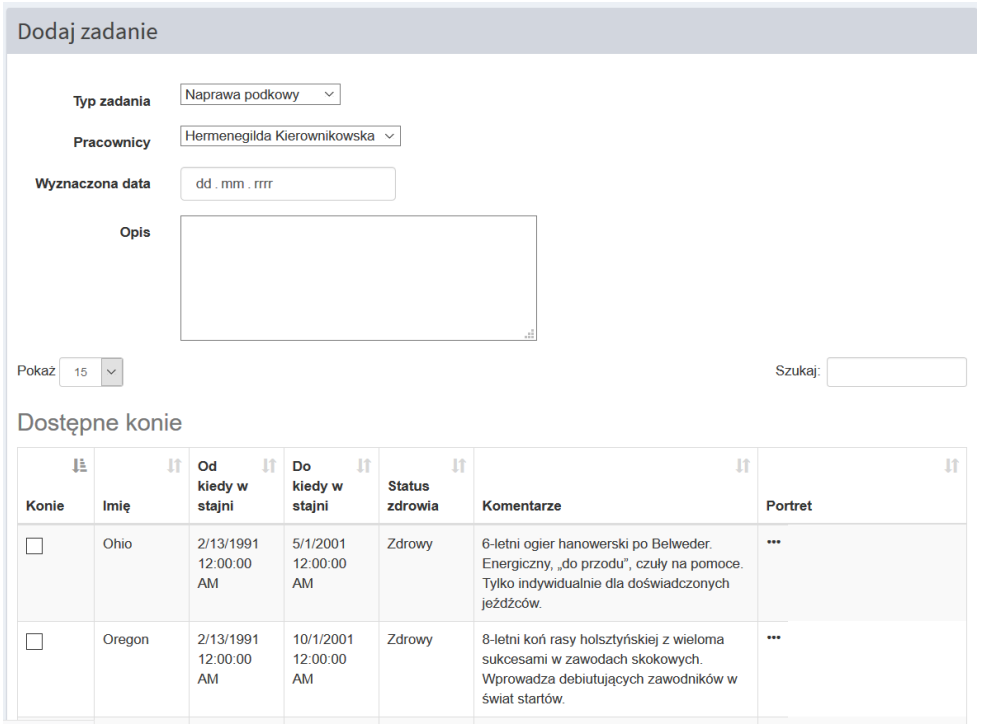
Rysunek 54. Dodaj konia. Źródło: Opracowanie własne.

Widok administracyjny widoczny na rysunku 55 przedstawia widok zawierający tabelę ze wszystkimi końmi, których dane są zapisane w systemie. Po kliknięciu w przycisk z trzema kropkami pokazują się opcje umożliwiające edycję konia, przejście do widoku szczegółowego, zmianę zdjęcia profilowego konia. Na potrzeby łatwego poruszania się po tabeli zaimplementowano sortowanie dla każdego pola.

Rysunek 55.Widok wszystkich koni. Źródło: Opracowanie własne.

Zadania

Kierownik jako osoba odpowiedzialna za przydzielanie zadań pracownikom, posiada dostęp do modułu dodawania zadań przedstawionego na rysunku 56.

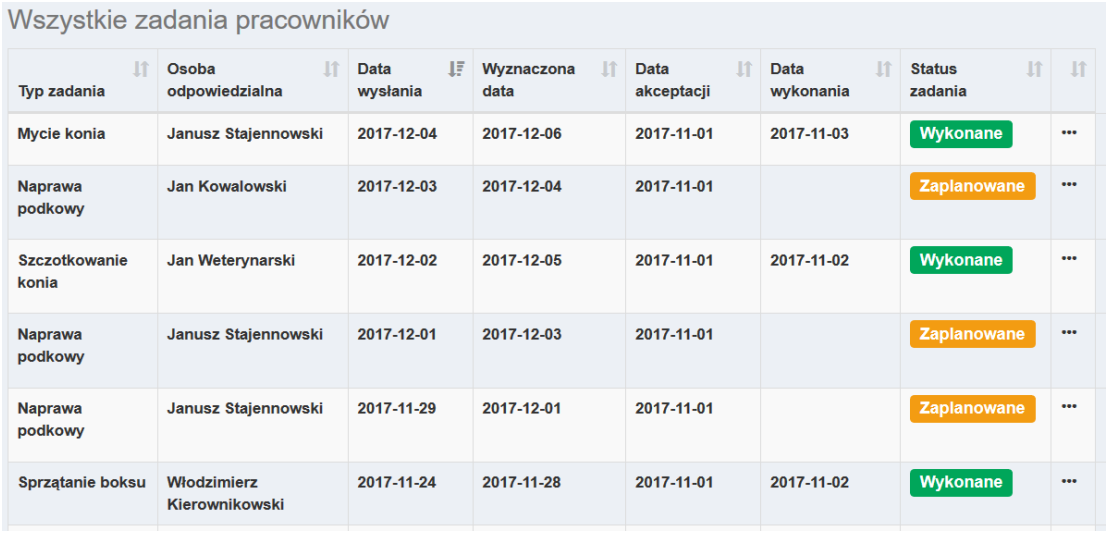


Rysunek 56. Dodaj zadanie. Źródło: Opracowanie własne.

Kierownik posiada dostęp do listy koni znajdujących się w systemie. W systemie SIDOS sprecyzowano typy zadań takie jak:

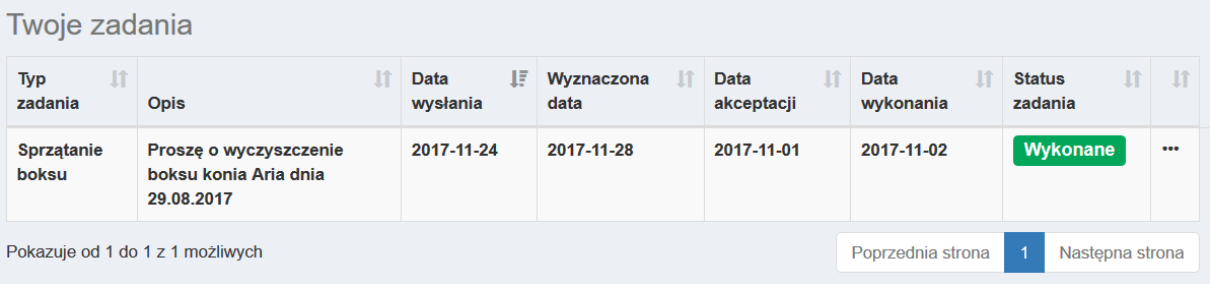
- Naprawa podkowy
- Karmienie
- Leczenie
- Mycie konia
- Sprzątanie boksu
- Szczotkowanie konia

Do danego typu zadania przypisywany jest zatrudnionego pracownika stadniny. Pracownicy znajdują się w liście rozwijanej. Każde zadanie zawiera datę wykonania oraz opis dotyczący szczegółów zadania. Widok dedykowany kierownikom znajdujący się na rysunku 57 przedstawia listę wszystkich zadań przypisanych do pracowników oraz informacje o statusie wykonania zadania. Na potrzeby widoku zaimplementowano sortowanie dla każdego pola.



Rysunek 57.Wszystkie zadania pracowników. Źródło: Opracowanie własne.

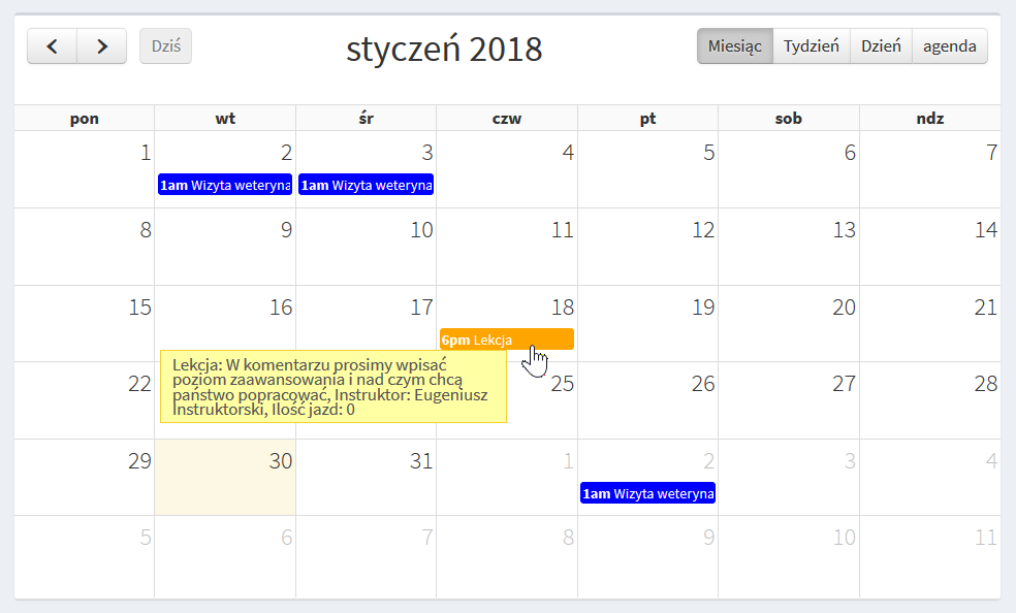
Pracownicy systemu SIDOS w widoku „Przeglądaj swoje zadania" posiadają dostęp do przypisanych zadań. Rysunek 58 prezentuje widok Przeglądaj zadania.



Rysunek 58. Widok Twoje zadania. Źródło: Opracowanie własne.

Kalendarz

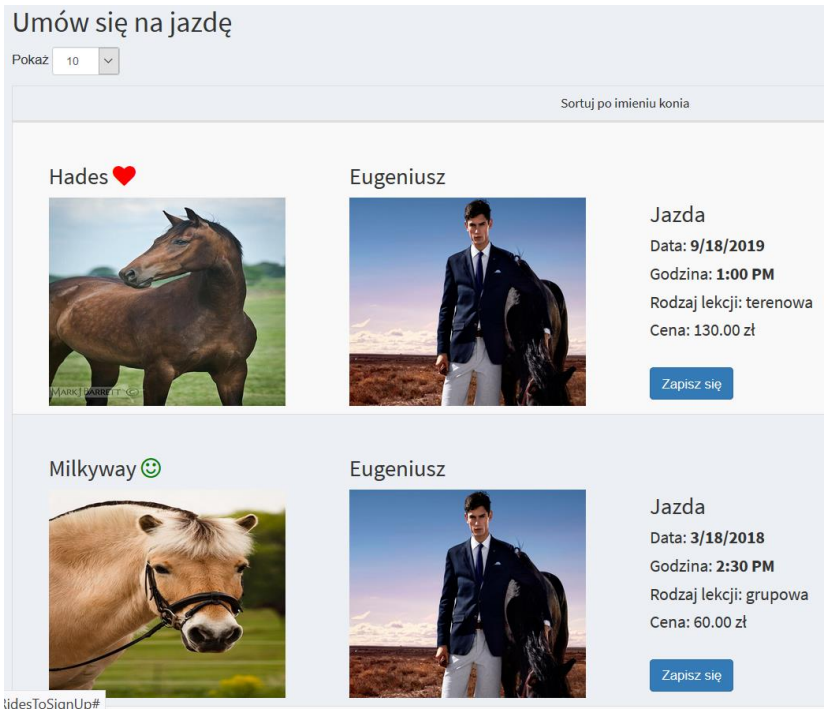
Jednym z podstawowych narzędzi do planowania i organizacji pracy jest kalendarz. Za jego pomącą widzimy graficznie uporządkowane zadania w czasie. Każdy pracownik posiada dostęp do harmonogramu swoich zadań. Rysunek 59 przedstawia kalendarz. Obsługo widoku jest niezwykle prosta po najechaniu kursorem na wydarzenie wyświetlany jest opis zadania.



Rysunek 59.Kalendarz. Źródło: Opracowanie własne.

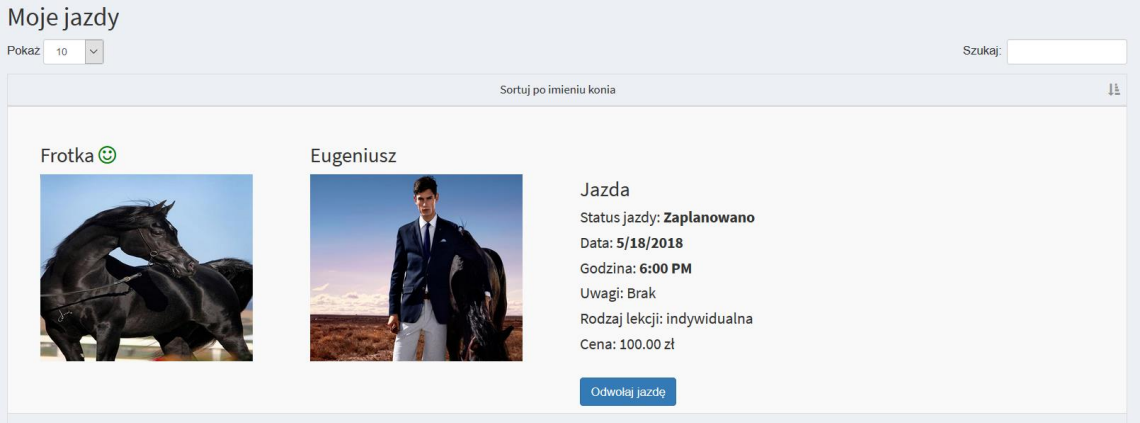
Zajęcia

Jedną z podstawowych funkcji systemu jest możliwość umówienia się na zajęcia jazdy konnej. Każdy zalogowany użytkownik może umówić się na jazdę poprzez widok znajdujący się na rysunku 60.



Rysunek 60.Umów się na jadze. Źródło: Opracowanie własne.

Spośród wielu koni o rożnym charakterze możemy wybrać konkretnego i umówić się na jazdę z instruktorem uwzględniając uwagi dotyczące jazdy. Zapisu na jazdę dokonujemy poprzez przycisk „zapisz się” znajdujący się na rysunku 60. Po zapisaniu się na jazdy w widoku „Wyświetl moje jazdy” posiadamy podgląd do aktualnych jazd oraz możliwość odwołania jazdy z podaniem powodu rezygnacji. Rysunek 61 przedstawia „Moje jazdy”.

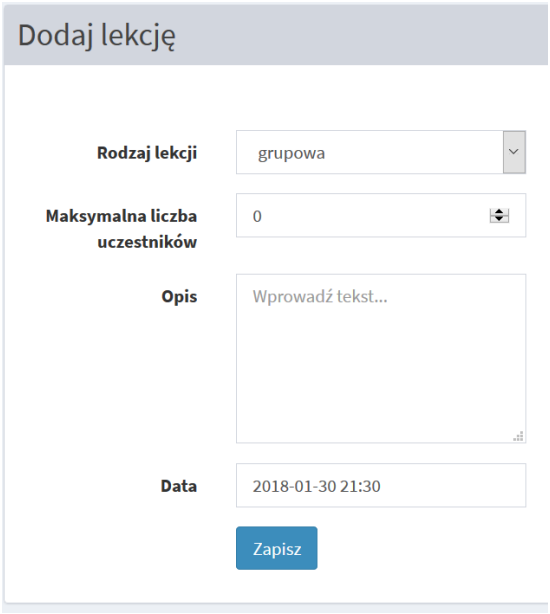


Rysunek 61.Moje jazdy. Źródło: Opracowanie własne.

Instruktor jako jedyny użytkownik w systemie posiada funkcje dodania lekcji, gdzie wyróżniono ich trzy rodzaje:

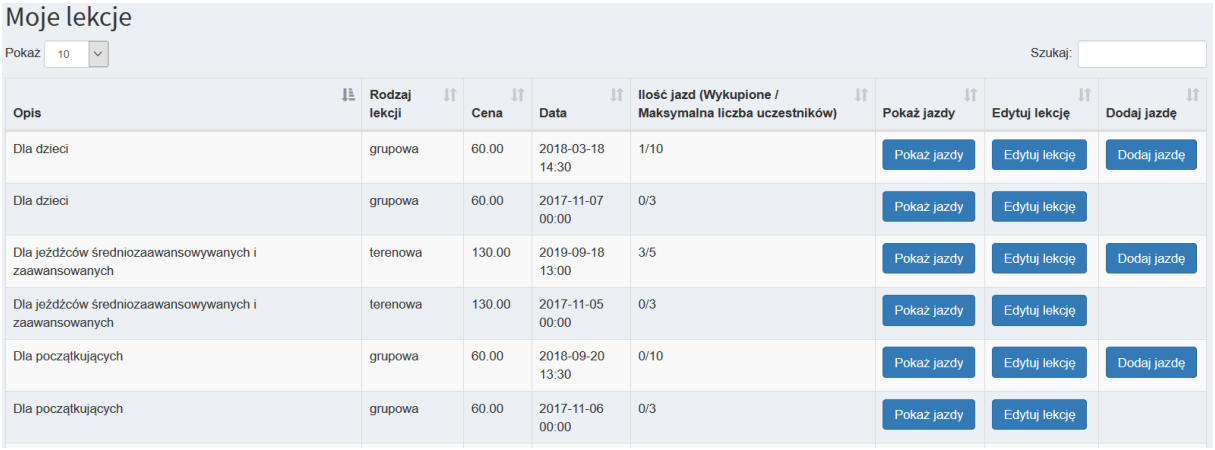
- Lekcja grupowa
- Lekcja terenowa
- Lekcja indywidualna

Na rysunku 62 znajduje się formularz umożliwiający określenie liczby osób, rodzaju lekcji oraz daty odbycia się lekcji.



Rysunek 62.Dodaj lekcje. Źródło: Opracowanie własne.

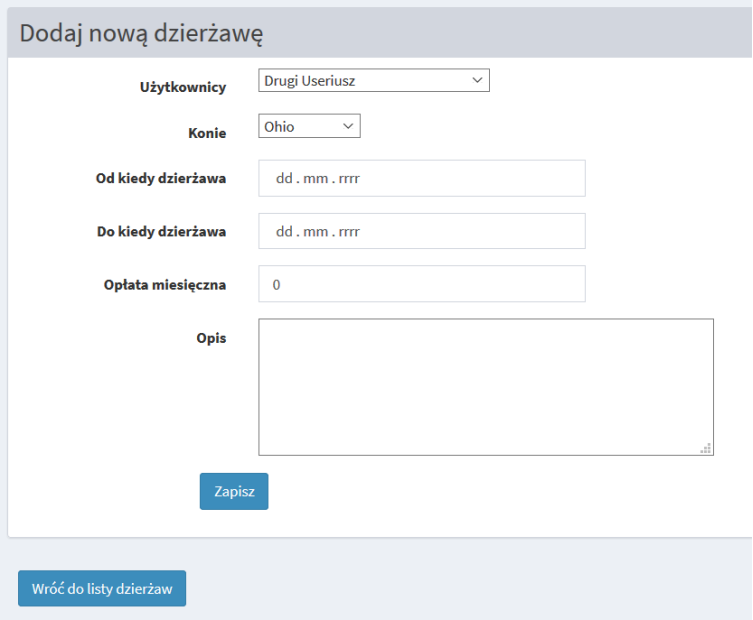
Panel przedstawiony na rysunku 63 „Wyświetl moje lekcje" pozwala instruktorowi na szybki podgląd lekcji. Dodatkowo zaimplementowano funkcje edytowania jazdy, dodania jazdy oraz pokaż jazdy.



Rysunek 63.Moje lekcje. Źródło: Opracowanie własne.

Dzierżawa

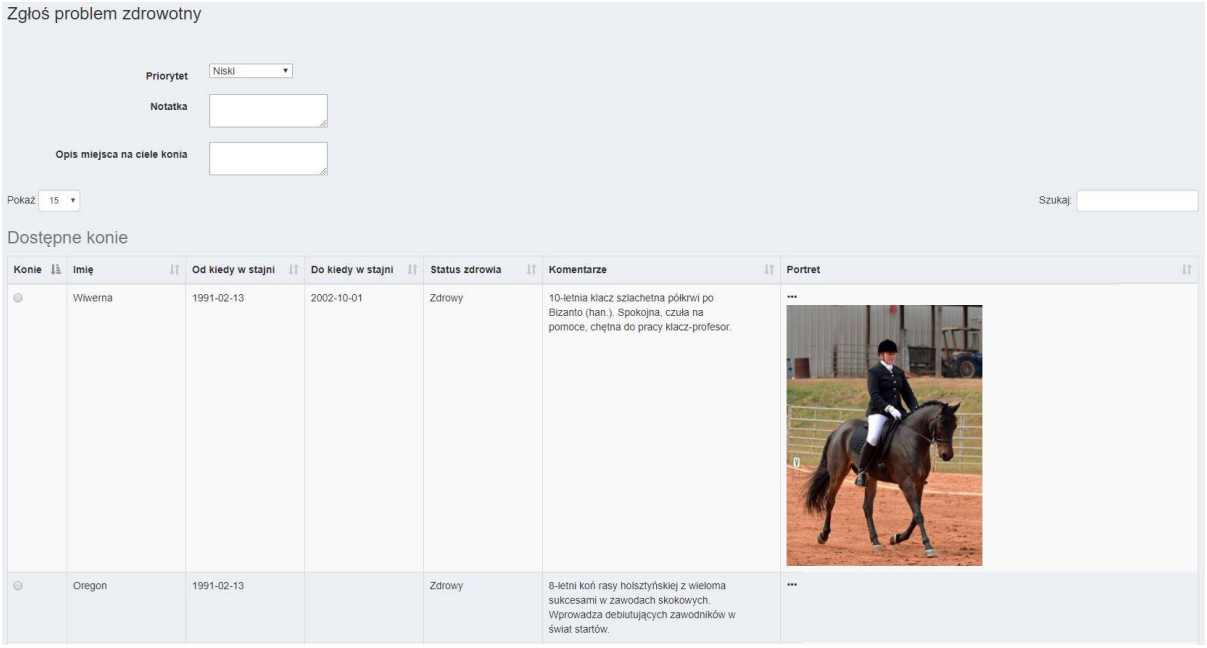
Funkcjonalnością wyróżniająca system SIDOS są dzierżawy koni. Zgłaszane są one do kierownika, który poprzez formularz ustala użytkownika oraz cenę dzierżawionego konia wraz z data rozpoczęcia i zakończenia dzierżawy, dodatkowo istnieje możliwość dodania opisu dzierżawy. Rysunek 64 przedstawia widok dodawania dzierżawy.



Rysunek 64.Dzierżawa.Źródło: Opracowanie własne.

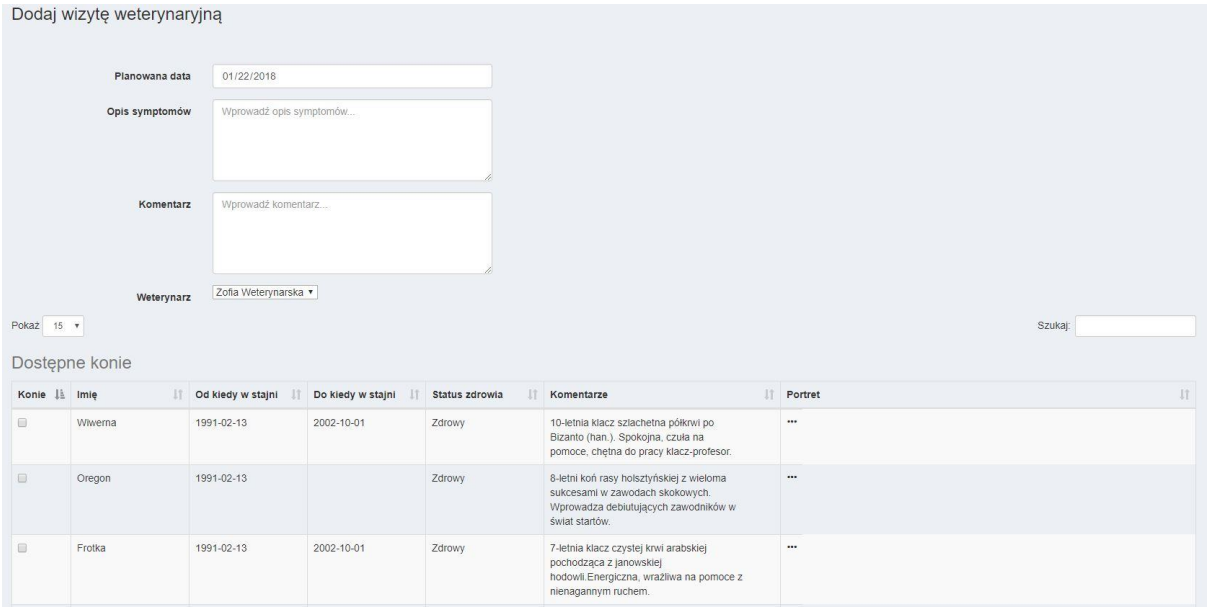
Wizyty weterynaryjne

W trosce o stan zdrowotny naszych konni w stadninie, system przewiduje zgłaszania problemu zdrowotnego konia oraz planowania wizyt weterynaryjnych. Wymienione funkcjonalności dedykowane są przede wszystkim weterynarzowi, który jest odpowiedzialny za utrzymanie konni w jak najlepszej kondycji zdrowotnej. System SIDOS został zaprojektowany z myślą o przechowywaniu historii wizyt weterynaryjnych. Rysunek 65 przedstawia formularz zgłoszenia problemu zdrowotnego.



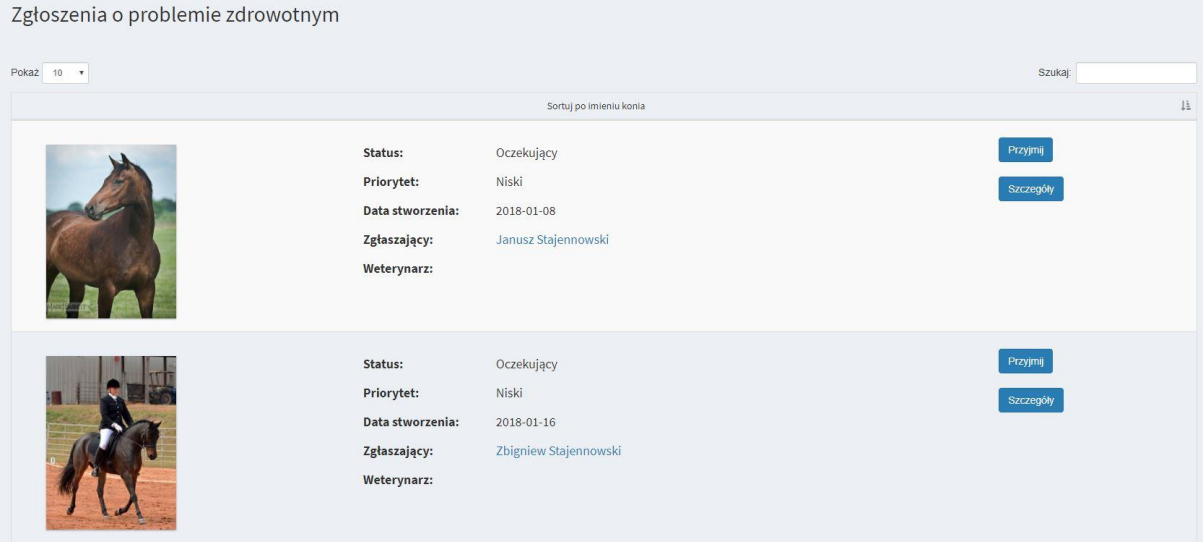
Rysunek 65. Zgłoś problem zdrowotny. Źródło: Opracowanie własne.

Wizytę weterynaryjną ustala pracownik, gdy zauważy niepokojące zachowanie u konia lub ranne. Realizacja wizyty odbywa się poprzez formularz, w którym wybieramy oraz weterynarza opisujemy symptomy chorego konia. Rysunek 66 przedstawia widok dodaj wizytę.



Rysunek 66. Dodaj wizytę weterynaryjna. Źródło: Opracowanie własne.

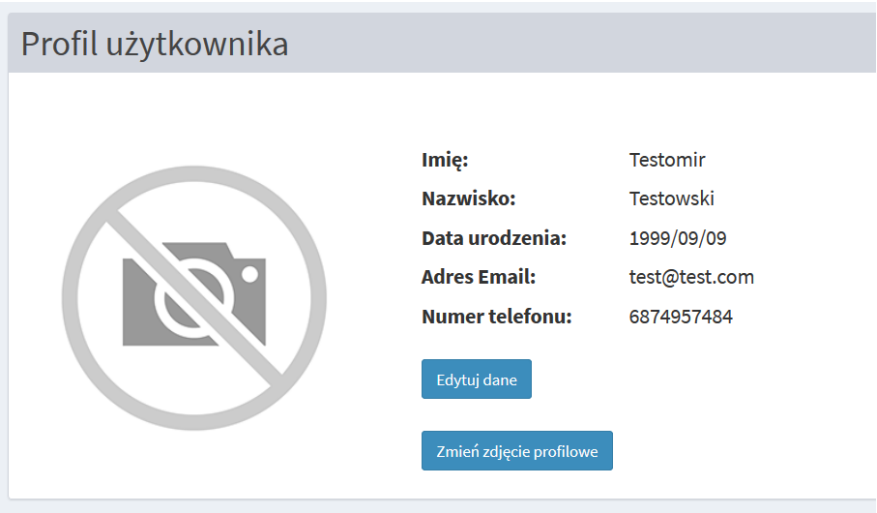
Weterynarz może przyjmować swoje zgłoszenia, które znajdują się w widoku przedstawionym na rysunku 67.



Rysunek 67.Zgłoszenie o problemie zdrowotnym. Źródło: Opracowanie własne.

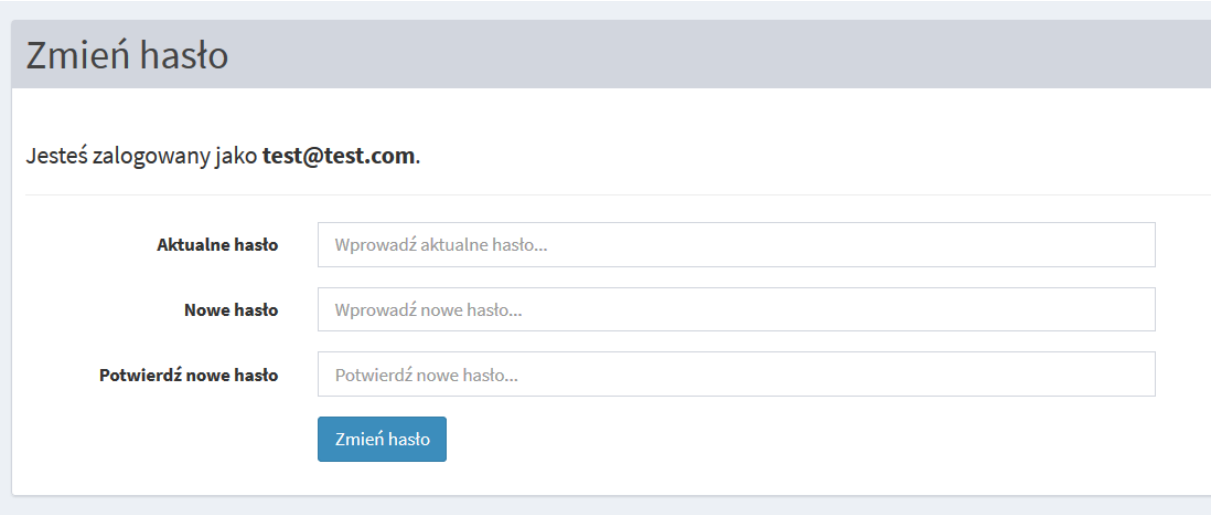
Zarządzaj profilem

Miejsce, w którym użytkownik systemu może edytować swój profil oraz wgrać nowe zdjęcie profilowe. Rysunek 68 przedstawia widok profilu użytkownika.



Rysunek 68.Profil użytkownika. Źródło: Opracowanie własne.

Rysunek 69 prezentuje opcje zmiany hasła.



Rysunek 69.Zmiana hasła. Źródło: Opracowanie własne.

5.3 Komunikacja z poziomu aplikacji

Komunikacja aplikacji mobilnej z bazą danych wykorzystuje architekturę Model-View-ViewModel. Model podzielony jest na trzy warstwy:

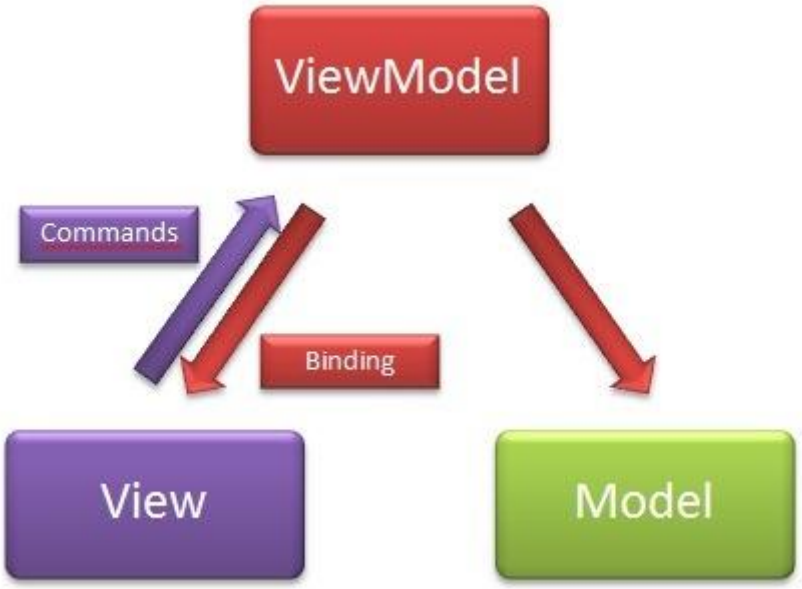
- View
- Model
- ViewModel

Warstwa Model przechowuje dane oraz obiekty tworzone w aplikacji. Jest to odzwierciedlenie jednej encji w bazie danych.

Warstwa View reprezentuje widok aplikacji. Do niej przekazywane są gotowe dane, warstwa ta służy do wyświetlania użytkownikowi przygotowanych danych oraz do przyjmowania od niego poleceń.

ViewModel jest warstwą, która odpowiada za porozumiewanie się z warstwą View oraz za oprawę danych przed wyświetleniem użytkownikowi.

Rysunek 70 ilustruje komunikację między wartstwami wzorca.



Rysunek 70.Architektura MVVM

Przesyłanie danych do aplikacji realizowane jest za pomocą metody POST, która jest oparta o protokół http, dzięki któremu możemy łączyć się z WebService'ami aplikacji. Pełnią one funkcję pobierania oraz przesyłania danych na lub z serwera aplikacji. Metoda GET, wykorzystująca protokół http, umożliwia pobieranie danych z serwera aplikacji web do docelowego serwera aplikacji mobilnej.

Listing 15 przedstawia metodę GET łączącą się za pomocą protokołu http. W odpowiedzi generuje się String, który jest pobierany z serwera Web aplikacji.

Listing 15.Metoda GET.

```
public static String sendGet (String url) throws Exception {

  URL obj = new URL(url);
  HttpURLConnection con = (HttpURLConnection) obj.opecConnection();

  con.setRequestProperty("Content-Type", "application/json:data=verbose");
  con.setRequestProperty("Accept", "application/json:data=verbose");
  if(Utils.UserTokenCls != null) {
      con.setRequest("Authorization", Utils.UserTokenCls.getToken_type() + "
" + Utils.UserTokenCls.getAccess_token());
  }
  //optional default is GET
  con.setRequestMethod("GET");

  BufferedReader in = new BufferedReader (
      new InputStreamReader(con.getInputStream()));
  String inputLine;
  StringBuffer response = new StringBuffer();

  while ((inputLine = in.readLine() ) != null) {
      response.append(inputLine):
  }
  in.close();
```

```
        //print result
        return response.toString();
      }
```

Listing 16 przedstawia metodę POST wysyłającą dane na serwer.

Listing 16.Metoda POST.

```
public static void
String sendPost (String url, String parameters) throws Exception {

  URL obj = new URL(url);
  HttpURLConnection con = (HttpURLConnection) obj.opecConnection();

  //add request header
  con.setRequestMethod("POST");

  con.setRequestProperty("Content-Type", "application/json:data=verbose");
  if(Utils.UserTokenCls != null) {
      con.setRequest("Authorization", Utils.UserTokenCls.getToken_type() + " " + Ut
ils.UderTokenCls.getAccess_token());
  }

  con.setDoOutput(true);
  DataOutputStream wr = new DataOutputStream(con.getPutputStream());
  wr.writeBytes(parameters);
  wr.flush();
  wr.close();

  int responseCode = con.getResponseCode();

  BufferedReader in = new BufferedReader (
      new InputStreamReader(con.getInputStream()));
  String inputLine;
  StringBuffer response = new StringBuffer();

  while ((inputLine = in.readLine() ) != null) {
      response.append(inputLine):
  }
  in.close();

  //print result
  System.out.println(response.toString());
  if(responseCode >= 400 && responseCode < 600) {
    throw new Exception();
  }
}
```

W listingu 16 dane są wysyłane na serwer, a w odpowiedzi z serwera przychodzi kod, który odpowiada za powodzenie albo za nie powodzenie operacji. Jeśli kod jest liczbą większą niż 400 i mniejszą niż 600 to znaczy, że operacja nie powiodła się. Inny wynik oznacza, że operacja została zakończona sukcesem. Dane są wysyłane w formacie JSON.

Autoryzacja i logowanie

Autoryzacja i logowanie odbywają się za pomocą Webservice, dzięki któremu następuje łączenie się z Webową aplikacją i pobieranie z niego tokena. Token ten jest

przypisany do konkretnego użytkownika i jest będzie od tej pory używany przy łączeniu się z Webservicem.

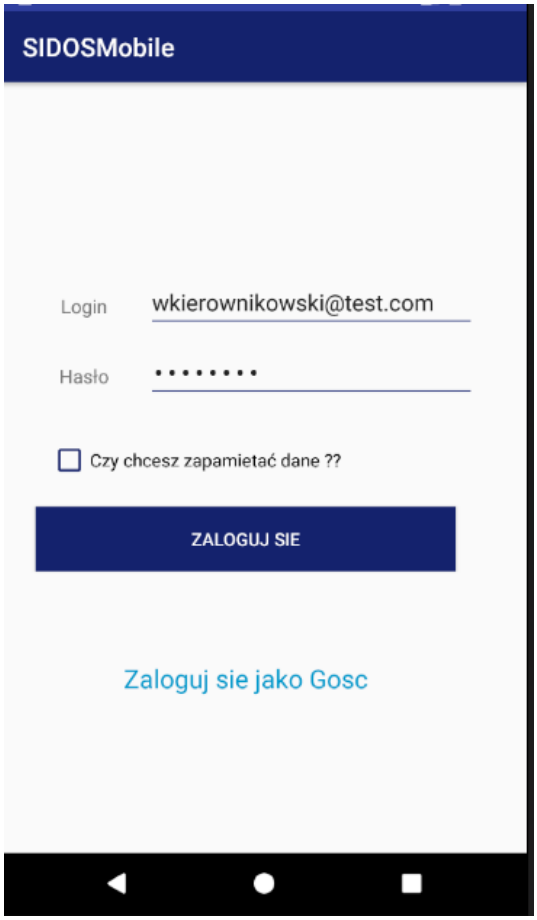
Na listingu 17 przedstawiono klasę, która odpowiada za obiekt trzymający token oraz pozostałe informacje takie jak:

Token_type - typ danego tokenu
userName - login użytkownika
EncryptedPass - szyfrowane hasło użytkownika

Listing 17. Token użytkownika

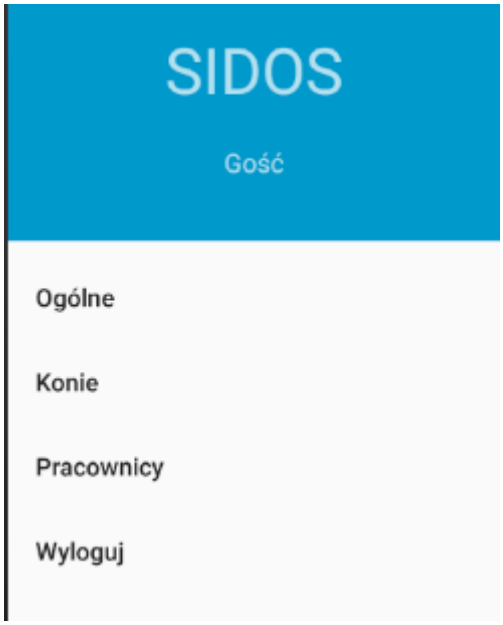
```
public class UserTokens implements Serializable{

  private String access_token;
  private String token_type;
  private String userName;
  private String EncryptedPass;

  public UserTokens(String access_token, String token_type, String userName
, String EncryptedPass) {
    this.access_token = access_token;
    this.token_type = token_type;
    this.userName = userName;
    this.EncryptedPass = EncryptedPass;
  }
}
```

Rysunek 71 przedstawia widok logowania z poziomu aplikacji mobilnej na której wyróżniono możliwość logowania jako pracownik oraz gość.



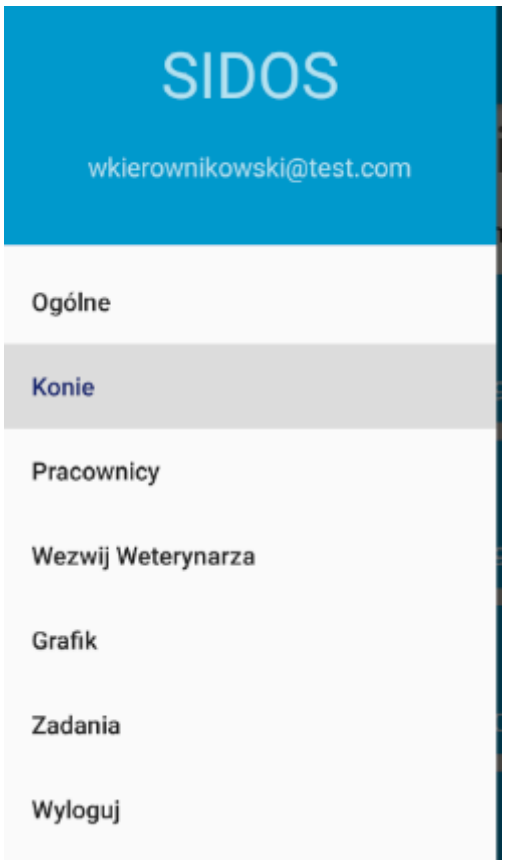
Rysunek 71.Widok logowania aplikacja mobilna. Źródło: Opracowanie własne.

Gość ma ograniczone prawa dostępu do funkcjonalności aplikacji: może przeglądać profile konia oraz pracownika. Rysunek 72 przedstawia funkcjonalności dla gościa.



Rysunek 72. Menu funkcjonalności gościa. Źródło: Opracowanie własne.

Rysunek 73 przedstawia funkcjonalności dostępne dla pracowników.



Rysunek 73. Menu funkcjonalności dla pracownika. Źródło: Opracowanie własne.

Przy logowaniu z poziomu aplikacji mobilnej zostaje wysłany request metodą POST na specjalny adres `/api/OAuth/Token`. W zawartości (body) requestu znajduje się login, hasło oraz typ dostępu (`gran_type`) enkodowane w formie `x-www-form-urlencoded`. Jeśli dane logowania są poprawne, w odpowiedzi przychodzi specjalny ciąg znaków (token) enkodowany w formie Base64, na podstawie którego identyfikowany jest użytkownik. Wraz z tokenem zwracana jest również informacja o roli użytkownika, dla którego wygenerowano token oraz inne informacje jak data wygenerowania, wygaśnięcia tokenu. Listing 18 przedstawia przykładową odpowiedź na pomyślne wysłanie zapytania o token.

Listing 18. Przykładowa odpowiedź na pomyślne wysłanie zapytanie o token w formacie JS

```
{
  "access_token": "-2ORcSGmBWMOWYwH8yJ3oFSDPmPtYINW8llZzp-
w_hzvla1rc_4q8ZgwRyv8euJNDtj11tkjlZU7ReMt0B3klb93TPPZlxiSGW0U-
tlf5DWKJbkT9fNGIJNviTX_o1WGpIQUZacrkve5Tl5BxPJvZKKcTYRQyTCXOiw5J2z0_lHYDibV
ZNFHV4JDL-
2pzHctYr2XjVMhC48V3UiEf8LrKr2HjYR_qCNaIJN99MwtbFxlBXE_ZeLo7RI0ShDUXMpo7HLCR
```

```
YXMEV0-G59NUJpabwn7OmdWCRXX1aMTUwj5QKD7NE0jCk-
vmZIrhlLmTRjW9mFlME5CeywB9zyNYpnELCwWsPw0CCUcuTCLuQt7RFFLJyfJMLjQdx4uT2Et4n
boKwaYl9mvWgRZoVHj0aZcCG2NpR7LFwBgiRWsHIESu6o_JwfBXGSpTQYC8t06fHxj1yLOKy8Gm
NSWTAxsQSK0vV0GNTXrLtjUbByR9DjWk_LI6YW9XKPxLkANH_iArdYf2_STBelaFR_HzB5R3t99
P4hm3c2cTzKu44PVjrrI8eA",
  "token_type": "bearer",
  "expires_in": 1209599,
  "userName": "wkierownikowski@test.com",
  "roles": "Supervisor",
  ".issued": "Sun, 04 Feb 2018 19:42:23 GMT",
  ".expires": "Sun, 18 Feb 2018 19:42:23 GMT"
}
```


5.4 Wsparcie dla pracowników ośrodka jeździeckiego

System SIDOS wspiera pracowników stajni w wykonywaniu codziennych obowiązków. Projektując aplikację wzięliśmy pod uwagę wszystkie stanowiska występujące w ośrodku jeździeckim.

Wsparcie dla kierownika

Projekt opisany w niniejszej pracy ułatwia pracę kierownika stajni, pozwalając mu w łatwy sposób tworzyć i przydzielać zadania pozostałym pracownikom, jak również weryfikować postęp prac oraz ich historię. Dodatkowo ewidencjonowanie koni znajdujących się w stajni stało się prostsze dzięki funkcjonalnościom naszego systemu, takim jak listowanie wszystkich zwierząt znajdujących się w stajni lub tylko tych, będących własnością ośrodka.

Aplikacja SIDOS przewiduje potrzeby pracowników niższego szczebla, dlatego zaimplementowane zostały funkcjonalności zwiększające wydajność ich pracy. W przypadku pracy wykonywanej przez stajennego, zgłoszenia zostały podzielone na najczęściej używane zagadnienia. Dzięki temu osoba zlecająca zadanie może wybrać z listy rodzaj pracy, przez co pojawiają nam się dedykowane parametry pozwalające je uszczegółowić jak np. dawkę karmy dla konia. Wybór zleceń dotyczących podkuwania koni i innych temu podobnych trafia w bezpośrednio do kowala. Tak samo jak w przypadku stajennego, pracownik może to zadanie zaakceptować, zrealizować, odmówić jego wykonania wcześniej podając powód rezygnacji. Każda akcja zostanie zarejestrowana wraz z datą i będzie do wglądu przez kierownika.

Wsparcie dla instruktorów

Instruktor ma do dyspozycji całą sekcję umożliwiającą zarządzanie lekcjami, a także poszczególnymi jazdami w nich zawartymi. Ma możliwość tworzenia, edycji, usuwania lekcji i jazd. Oprócz tego może przypisać konia do konkretnych zajęć. Każdy trener może pochwalić się certyfikatami, które zdobył w trakcie swojej kariery, dzięki funkcjonalności, która umożliwia instruktorom dodawanie certyfikatów do swojego profilu. Umożliwia to klientom analizę profesjonalizmu pracownika względem swoich potrzeb.

Wsparcie dla weterynarzy

Bardzo ważnym elementem każdej sprawnie działającej stadniny jest weterynarz. Zwierzęta znajdujące się w stajni wymagają opieki lekarza, z racji ich dużej liczby, jak również faktu, że część z nich jest używana do startów w zawodach jeździeckich. W związku z tym konie muszą być zawsze do dyspozycji, zdrowe. Jeśli jednak przytrafiłby się im jakiś wypadek losowy np. upadek czy choroba, zadaniem zarówno stajni jak i weterynarza jest jak najszybsze doprowadzenie zwierząt do pełnego wyleczenia Aby zapewnić szybką diagnozę i skuteczną rekonwalescencję, w systemie SIDOS zaimplementowany został szereg funkcjonalności dostępnych dla weterynarza umożliwiający jak najszybszą reakcję na zgłoszenie. Każdy pracownik stajni może zgłosić problem zdrowotny, dzięki temu, jeśli np. koń podczas przejażdżki doznał kontuzji, instruktor natychmiast tworzy zgłoszenie, które trafia do weterynarza. Ten zaś, na podstawie szczegółów zawartych w zleceniu, takich jak: zdjęcie, opis, data, miejsce urazu jest w stanie przyjechać do stajni z odpowiednimi narzędziami, tak szybko jak to możliwe. Dodatkowo istnieje możliwość umówienia się na wizytę weterynaryjną, która nie wymaga pilnej obsługi, a przeprowadzana jest np. jedynie kontrolnie, co pewien okres.

6 Opis fazy testów

Testowanie jest nieodłącznym elementem wytwarzania oprogramowania. Ma na celu sprawdzić, czy system jest zgodny z założeniami i spełnia wszystkie postawione mu wymagania [38]. Testy dzielimy na 4 rodzaje:

1. Testy modułowe (unit/component testing)
2. Testy integracyjne (integration testing)
3. Testy systemowe (system testing)
4. Testy akceptacyjne (acceptance testing) [39]

Testy modułowe (jednostkowe) mają na celu sprawdzenie działania pojedynczych elementów kodu. Programista, za pomocą wcześniej przygotowanych danych, niezależnie od reszty aplikacji sprawdza poprawność poszczególnych metod. Wynik testu porównywany jest z wynikiem oczekiwanym. W przypadku zgodności wyników test uznawany jest za pozytywny, w innym przypadku - negatywny. Testy jednostkowe dzielą się na 4 typy:

a. Analizę ścieżek - programista określa punkt początkowy i końcowy dla testów i sprawdza wszystkie możliwe ścieżki między tymi punktami.
b. Wykorzystanie klas równoważności – są to zbiory danych wykorzystywanych podczas przeprowadzania testu. Jeżeli test daje wynik pozytywny dla kilku elementów zbioru, klasa zostaje uznana za poprawną bez konieczności testowania wszystkich jej elementów.
c. Testowanie wartości brzegowych – jest to przeprowadzenie testów z wykorzystaniem skrajnych wartości należących do danej klasy równoważności lub tych wychodzących poza klasę. Np. test wartości brzegowych dla zbioru [1-100] wykorzystywałby wartości 0, 1, 2, 99, 100, 101.
d. Testowanie składniowe – sprawdzenie czy dane wchodzące do systemu poddawane są walidacji. [40]

Testy integracyjne polegają na jednoczesnym przetestowaniu kilku elementów systemu. Mają na celu sprawdzenie działania zależnych od siebie komponentów razem. Testy te dzielą się na dwa rodzaje:

a. Testy między modułami – *„wykonywane w celu wykrycia usterek w interfejsach i interakcjach pomiędzy integrowanymi modułami”*[5]

[5] http://glossary.istqb.org/search/integration, data dostępu: 19.01.2018

b. Testy między systemami - *„weryfikują poprawne wykonanie komponentów oprogramowania i właściwe połączenie między komponentami w ramach rozwiązania.”*[6]

Przykładowo podczas testów jednostkowych połączenie z bazą danych jest symulowane przez programistę poprzez ręczne wpisywanie danych. Dane są następnie przetwarzane i metoda zwraca określony wynik. Podczas testów integracyjnych realizowane jest faktyczne połączenie z bazą. Metoda pobiera z niej dane, a następnie przetwarza i zwraca określony rezultat, jak w przypadku testów jednostkowych. [41]

Testy systemowe mają na celu sprawdzenie zgodności systemu z wymaganiami postawionymi podczas fazy analitycznej projektu. Testy przeprowadzane są z użyciem techniki czarnej skrzynki. Oznacza to, że osoba testująca nie musi mieć żadnej wiedzy o kodzie testowanej aplikacji. Podstawą testów są wymagania funkcjonalne, niefunkcjonalne oraz przypadki użycia. Środowisko testowe musi jak najwierniej odwzorować środowisko produkcyjne, aby zminimalizować ryzyko niewykrycia błędu z powodu różnicy wykorzystywanych środowisk. [42][43]

Testy akceptacyjne są ostatnim etapem przeprowadzania testów. Testerem w tym przypadku jest klient, który sprawdza, czy produkt końcowy jest zgodny z ustalonymi wymaganiami. Jeżeli system spełnia wszystkie założenia zostaje uruchomiony na środowisku produkcyjnym.

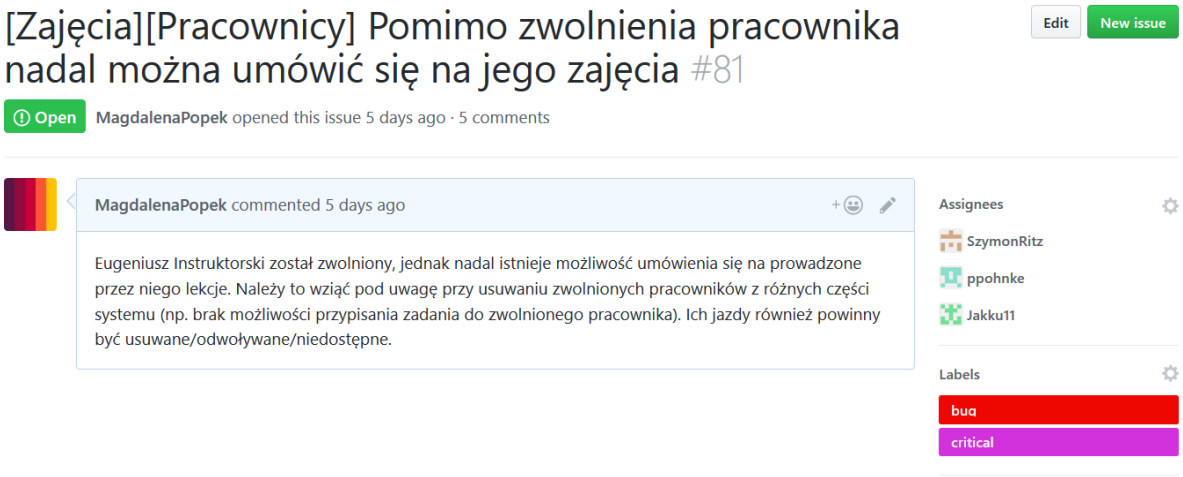
W trakcie tworzenia systemu SIDOS przeprowadzane były testy systemowe. Grupa testowa, w skład której wchodzili:

1. Magdalena Popek (Lider zespołu testów)
2. Paulina Gruba
3. Paweł Walesiak
4. Jacek Muszyński

przeprowadzała testy z użyciem scenariuszy testowych. Na zakończenie każdej iteracji testów aplikacji grupa testowa sporządzała listę błędów w postaci zgłoszeń na platformie programistycznej GitHub. Zgłoszenia były opisywane w sposób pozwalający odtworzyć napotkany błąd, a funkcjonalność tagowania pozwalała w prosty sposób przekazać informację o wadze i naturze problemu. Każde zgłoszenie zawierało odpowiednio sformułowany tytuł

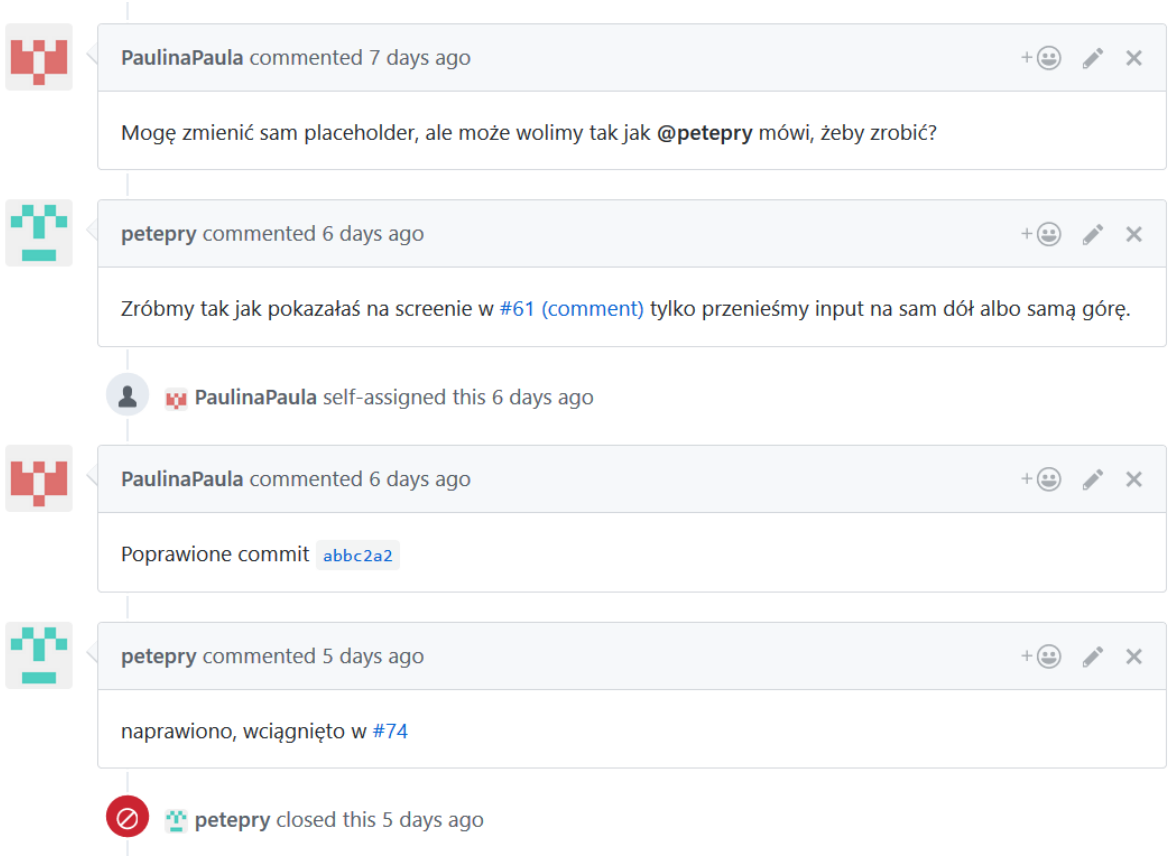
[6] http://istqbexamcertification.com/what-is-system-integration-testing/, data dostępu: 19.01.2018

ułatwiający szybką identyfikację, treść opisującą problem oraz załączniki, np. zrzuty ekranu z wadliwym elementem. Rysunek 74 przedstawia przykładowe zgłoszenie o błędzie.



Rysunek 74. Przykładowe zgłoszenie o błędzie wystawione na platformie GitHub przez członka zespołu testów.

Funkcja komentarzy przedstawiona na rysunku 75 umożliwiała doprecyzowanie błędów, odniesienie się innych osób do zgłoszenia, poinformowanie o stanie prac lub wymianę pomysłów na jak najlepsze rozwiązanie zaistniałego problemu.



Rysunek 75. Komentarze pod jednym ze zgłoszeń o błędzie.

Tabele numer 14 do 27 przedstawiają przykładowe scenariusze testowe wykorzystywane przez grupę testów.

Tabela 14. Scenariusz testowy aplikacji webowej "Umów się na lekcję".

P1	Umów się na lekcję
Scenariusz dotyczy	Umówienia się na lekcję przez klienta
Cel testu	Sprawdzenie poprawności działania funkcjonalności umawiania się na lekcje
Sposób dostępu	Poprzez konto zalogowanego użytnowknika
Scenariusz (kroki testowe)	
Warunek początkowy	Użytkownik zalogowany na konto klienta
Akcje użytkownika	Odpowiedź systemu
1. Wejście w zakładkę „Zajęcia” w menu głównym 3. Wejście w opcję „Umów się na jazdę” 5. Wybranie jazdy, którą klient jest zainteresowany i kliknięcie przycisku	2. System wyświetla opcje menu „Zajęcia” 4. System wyświetla stronę umawiania się na jazdę 6. System wyświetla stronę z polem tekstowym do wpisywania uwag dotyczących lekcji

„Zapisz się” 7. Klient wpisuje uwagi dotyczące jazdy (nieobowiązkowe) i klika „Zapisz się”	8. System przekierowuje klienta na stronę „Moje jazdy”
Ocena testu	Pozytywna

Tabela 15. Scenariusz testowy aplikacji webowej "Wyślij wiadomość".

P3	Wyślij wiadomość
Scenariusz dotyczy	Wysyłania wiadomości z konta zalogowanwgo użytkownika
Cel testu	Sprawdzenie poprawności działania funkcjonalności wysyłania wiadomości
Sposób dostępu	Poprzez konto zalogowanego klienta
Scenariusz (kroki testowe)	
Warunek początkowy	Użytkownik zalogowany na konto klienta
Akcje użytkownika	Odpowiedź systemu
1. Wejście w zakładkę „Wiadomości” poprzez kliknięcie w ikonę koperty 3. Wejście w opcję „Nowa wiadomość” 5. Wpisanie nieistniejącego adresu email 7. Wpisanie poprawnego adresu email, dowolnego tematu, treści wiadomości i kliknięcie „Wyślij” 9. W wiadomościach wysłanych widnieje wysłana wiadomość.	2. System wyświetla opcje menu „Wiadomości” 4. System wyświetla formularz nowej wiadomości 6. System wyświetla komunikat o braku poprawności danych 8. System przekierowuje klienta do „Wiadomości wysłane”
Ocena testu	Pozytywna

Tabela 16. Scenariusz testowy aplikacji webowej "Odbierz wiadomość".

P4	Odbierz wiadomość
Scenariusz dotyczy	Funkcjonalności odbierania wiadomości
Cel testu	Sprawdzenie poprawności działania funkcjonalności odbierania wiadomości
Sposób dostępu	Poprzez konto zalogowanego użytkownika – odbiorcy wiadomości
Scenariusz (kroki testowe)	
Warunek początkowy	Użytkownik zalogowany na konto odbiorcy wiadomości
Akcje użytkownika	Odpowiedź systemu
1. Wejście w zakładkę „Wiadomości” poprzez kliknięcie w ikonę koperty 3. Wybranie opcji „Skrzynka odbiorcza” 5. Odnalezienie wysłanej przez klienta wiadomości i wejście w „Szczegóły”	2. System wyświetla opcje menu „Wiadomości” 4. System przekierowuje użytkownika do skrzynki odbiorczej 6. System wyświetla szczegóły wiadomości 8. System przekierowuje użytkownika do skrzynki odbiorczej

7. Odczytanie szczegółów wiadomości i kliknięcie „Powrót”	
Ocena testu	Pozytywna

Tabela 17. Scenariusz testowy aplikacji webowej "Dodaj dzierżawę".

P5	Dodaj dzierżawę
Scenariusz dotyczy	Przypisania konia w dzierżawę
Cel testu	Sprawdzenie możliwości utworzenia dzierżawy
Sposób dostępu	Poprzez konto zalogowanego kierownika
Scenariusz (kroki testowe)	
Warunek początkowy	Użytkownik zalogowany na konto kierownika
Akcje użytkownika	Odpowiedź systemu
1. Wejście w zakładkę „Dzierżawa” w menu głównym 3. Wejście w opcję „Dodaj dzierżawę” 5. Wybranie użytkownika, który bierze konia w dzierżawę, wybór konia do dzierżawy, wprowadzenie daty początku dzierżawy, pominięcie daty końca dzierżawy i komentarza. Kliknięcie „Zapisz” 7. Dodana dzierżawa widoczna jest w tabeli	2. System wyświetla opcje menu „Dzierżawa” 4. System wyświetla formularz przypisywania konia do dzierżawy 6. System wyświetla stronę z polem tekstowym do wpisywania uwag dotyczących lekcji 8. System przekierowuje kierownika do widoku „Dzierżawy koni”
Ocena testu	Pozytywna

Tabela 18. Scenariusz testowy aplikacji webowej "Dodaj pracownika".

P6	Dodaj pracownika
Scenariusz dotyczy	Dodanie nowego pracownika do systemu
Cel testu	Sprawdzenie poprawności dodawnaia nowych pracwników
Sposób dostępu	Poprzez konto zalogowanego kierownika
Scenariusz (kroki testowe)	
Warunek początkowy	Użytkownik zalogowany na konto kierownika
Akcje użytkownika	Odpowiedź systemu
1. Wejście w zakładkę „Zarządzaj stajnią” w menu głównym 3. Wejście w opcję „Dodaj pracownika” 5. Kliknięcie „Zapisz” bez wprowadzania danych 7. Wypełnienie pól z danymi oraz kliknięcie „Zapisz” 9. Dodany pracownik widnieje na liście. Wejście z zakładkę „O nas” oraz przejście do zakładki „Pracownicy” 11. Dodany pracownik wyświetlany	2. System wyświetla opcje menu „Zarządzaj stajnią” 4. System wyświetla formularz dodawania pracownika 6. System wyświetla komunikaty pod każdym z pól o konieczności wypełnienia go 8. System przekierowuje użytkownika na stronę „Wszyscy pracownicy” 10. System wyświetla widok z pracownikami 12. System wyświetla szczegółowy profil pracownika

jest w widoku „Pracownicy”. Wejście w „Szczegóły”	
Ocena testu	Pozytywna

Tabela 19. Scenariusz testowy aplikacji webowej "Dodaj konia".

P7	Dodaj konia
Scenariusz dotyczy	Dodanie nowego konia do systemu
Cel testu	Sprawdzenie poprawności dodawnaia konia
Sposób dostępu	Poprzez konto zalogowanego kierownika
Scenariusz (kroki testowe)	
Warunek początkowy	Użytkownik zalogowany na konto kierownika
Akcje użytkownika	Odpowiedź systemu
1. Wejście w zakładkę „Zarządzaj stajnią” w menu głównym 3. Wejście w opcję „Dodaj konia” 5. Kliknięcie „Zapisz” bez wprowadzania danych 7. Wypełnienie pól z danymi oraz kliknięcie „Zapisz” 9. Dodany koń widnieje na liście. Wejście z zakładkę „O nas” oraz przejście do zakładki „Konie” 11. Dodany koń wyświetlany jest w widoku „Konie”. Wejście w „Szczegóły”	2. System wyświetla opcje menu „Zarządzaj stajnią” 4. System wyświetla formularz dodawania konia 6. System wyświetla komunikaty pod polami wymaganymi o konieczności wypełnienia ich 8. System przekierowuje użytkownika na stronę „Wszystkie konie” 10. System wyświetla widok z końmi 12. Wystąpił błąd aplikacji „Value cannot be null or empty”
Ocena testu	Negatywna

Tabela 20. Scenariusz testowy aplikacji mobilnej "Zgłoszenie do weterynarza".

P1	Zgłoszenie do weterynarza
Scenariusz dotyczy	Utworzenie nowego zgłoszenia do weterynarza
Cel testu	Sprawdzenie poprawności działania funkcjonalności zgłoszenia do weterynarza
Sposób dostępu	Poprzez konto zalogowanego kierownika
Scenariusz (kroki testowe)	
Warunek początkowy	Użytkownik zalogowany na konto kierownika
Akcje użytkownika	Odpowiedź systemu
1. Wejście w zakładkę „Wezwij weterynarza” w menu głównym 3. Wejście w opcję „Zgłoś” 5. Wypełnienie formularzu, wybranie konia, weterynarza, wpisanie lokalizacji uszczerbku, wybranie priorytetu z listy oraz wgranie zdjęcia i dodanie opisu. Użytkownik klika przycisk „Wyślij”.	2. System wyświetla opcje menu „Wezwij weterynarza” 4. System wyświetla widok formularzu „Zgłoś” 6. System wyświetla błąd „Nie można dodać zgłoszenia”
Ocena testu	Negatywna

Tabela 21. Scenariusz testowy aplikacji mobilnej "Wybranie z listy zgłoszenia do weterynarza".

P2	Wybranie z listy zgłoszenia do weterynarza
Scenariusz dotyczy	Wyboro zgłoszenia do weterynarza z listy
Cel testu	Sprawdzenie poprawności działania funkcjonalności wyświetlania zgłoszeń
Sposób dostępu	Poprzez konto zalogowanego kierownika
Scenariusz (kroki testowe)	
Warunek początkowy	Użytkownik zalogowany na konto kierownika
Akcje użytkownika	Odpowiedź systemu
1. Wejście w zakładkę „Wezwij weterynarza” w menu głównym 3. Wybranie zgłoszenia z listy bez wartości pola „Zgłoszone przez”.	2. System wyświetla opcje w menu „Wezwij weterynarza” 4. System wyłącza aplikacje.
Ocena testu	Negatywna

Tabela 22. Scenariusz testowy aplikacji mobilnej "Wyloguj się".

P3	Wylogowanie klienta z aplikacji
Scenariusz dotyczy	Przycisku „Wyloguj się”
Cel testu	Sprawdzenie poprawności działania funkcjonalności wylogowywania
Sposób dostępu	Poprzez konto zalogowanego klienta
Scenariusz (kroki testowe)	
Warunek początkowy	Użytkownik zalogowany na konto klienta
Akcje użytkownika	Odpowiedź systemu
1.Wejście w zakładkę „Wyloguj się” w menu głównym	2. System przechodzi do panelu logowania
Ocena testu	Pozytywna

Tabela 23. Scenariusz testowy aplikacji mobilnej "Widok Ogólne".

P4	Widok Ogólne
Scenariusz dotyczy	Wyboru zakładki „Ogólne”
Cel testu	Sprawdzenie poprawności językowej
Sposób dostępu	Poprzez konto zalogowanego kierownika
Scenariusz (kroki testowe)	
Warunek początkowy	Użytkownik zalogowany na konto kierownika
Akcje użytkownika	Odpowiedź systemu
1. Wejście w zakładkę „Ogólne” w menu głównym	2. System wyświetla widok „Ogólne”. Opis tekstowy nie zawiera polskich znaków
Ocena testu	Negatywna

Tabela 24. Scenariusz testowy aplikacji mobilnej "Kryteria wyszukiwania konia".

P5	Kryteria wyszukiwania konia
Scenariusz dotyczy	Funkcji wyszukiwania koni

Cel testu	Sprawdzenie poprawności językowej oraz działania funkcjonalności wyszukiwania
Sposób dostępu	Poprzez konto kierownika
Scenariusz (kroki testowe)	
Warunek początkowy	Użytkownik zalogowany na konto kierownika
Akcje użytkownika	Odpowiedź systemu
1. Wejście w zakładkę „Konie” w menu głównym. 3. Wybór kryterium, po którym chcemy wyszukiwać.	1. System wyświetla listę koni 4. Kryteria wyszukiwania w języku angielskim
Ocena testu	Negatywna

Tabela 25. Scenariusz testowy aplikacji mobilnej "Zadania".

P6	Zadania
Scenariusz dotyczy	Funkcji widoku szczegółowego zadania
Cel testu	Sprawdzenie poprawności językowej
Sposób dostępu	Poprzez konto kierownika
Scenariusz (kroki testowe)	
Warunek początkowy	Użytkownik zalogowany na konto kierownika
Akcje użytkownika	Odpowiedź systemu
1. Wejście w zakładkę „Zadania” w menu głównym. 3.Wybór widoku szczegółowego zadania	2.System wyświetla listę zadań 4.Typ zadania jest wyświetlany po angielsku.
Ocena testu	Negatywna

Tabela 26. Scenariusz testowy aplikacji mobilnej "Lista pracowników".

P7	Lista pracowników
Scenariusz dotyczy	Wybór widoku „Pracownicy”
Cel testu	Sprawdzenie poprawności widoku
Sposób dostępu	Poprzez konto zalogowanego kierownika
Scenariusz (kroki testowe)	
Warunek początkowy	Użytkownik zalogowany na konto kierownika
Akcje użytkownika	Odpowiedź systemu
1. Wejście w zakładkę „Pracownicy” w menu głównym 3. Wybór profilu pracownika	2. System wyświetla listę pracowników 4. Wyświetla profil pracownika bez wartości pola daty
Ocena testu	Negatywna

Tabela 27. Scenariusz testowy aplikacji mobilnej "Lista koni".

P8	Lista koni
Scenariusz dotyczy	Wybór widoku „Konie”
Cel testu	Sprawdzenie poprawności językowej
Sposób dostępu	Poprzez konto zalogowanego kierownika
Scenariusz (kroki testowe)	

Warunek początkowy	Użytkownik zalogowany na konto kierownika
Akcje użytkownika	Odpowiedź systemu
1. Wejście w zakładkę „Konie” w menu głównym 3. Wybrać profil konia.	2. Wyświetla listę koni 4. Profil konia, posiada pole płeć w języku angielskim.
Ocena testu	Negatywna

7 Podsumowanie

W rozdziale tym zostaną przedstawione trudności, jakie napotkaliśmy podczas tworzenia systemu SIDOS oraz proponowane przez nas kierunki rozwoju.

7.1 Zrealizowane założenia

W projekcie udało się zrealizować większość elementów zaproponowanych w trakcie analizy wymagań. Zapewniono możliwość umawiania się na jazdy od strony klienta jak również anulowania spotkania, stworzyliśmy funkcjonalność przeglądania zarówno koni jak i pracowników stajni, powodując, że klient zawsze znajdzie odpowiedź na nurtujące go pytania.

Każdy zarejestrowany użytkownik w systemie ma możliwość złożenia wniosku o dzierżawę dotyczącego koni znajdujących się na liście aktualnie dostępnych do wynajmu. Dodatkowo zaimplementowano szereg funkcjonalności pozwalających na sprawne zarządzanie końmi ich właścicielom lub osobom je dzierżawiącym.

Z poziomu zakładki Moje konie możne edytować dane zwierzęcia jak również decydować o jego widoczności w systemie.

W systemie możliwe jest umówienie wizyty weterynaryjnej jak i zgłoszenie problemu zdrowotnego konia. Zadanie takie trafiają do weterynarzy, którzy podejmują dalsze działania mające na celu przywrócenie zwierząt do stanu zdrowia.

Projekt SIDOS wspiera także innych pracowników stajni jak np. stajennego i kowala pozwalając im realizować zadania, które są wyświetlane w systemie wraz ze wszystkimi niezbędnymi danymi niezbędnymi do poprawnego ich wykonania.

Zarządzanie dużą ilością koni, pracowników i zadań jest ogromnym wyzwaniem, więc w systemie zaimplementowano narzędzia pozwalające kierownikowi na zlecanie, przeglądanie zadań zleconych każdemu pracownikowi stajni. Przełożony ma również do dyspozycji listę wszelkich koni znajdujących się w stajni wraz z ich szczegółami co pozwala na sprawniejsze i wydajniejsze zarządzanie obiektem.

Zadania, jazdy, lekcje, wizyty weterynaryjne poszczególnych użytkowników wyświetlają się im w przystępny sposób na kalendarzu.

7.2 Napotkane trudności

Rozbudowane projekty często wiążą się z różnymi problemami, które utrudniają ich zrealizowanie. Jednak nie zawsze są to błędy i trudności związane w całości z programowaniem. Nierzadko wiążą się one jeszcze z projektowaniem systemu i współpracą w zespole. Trudności jakie napotkaliśmy w trakcie tworzenia systemu możemy podzielić na 3 rodzaje: projektowe, organizacyjne i techniczne.

7.2.1 Trudności projektowe

W trakcie fazy analizy i projektowania nie wszystko jest możliwe do przewidzenia. Wynika to głównie z braku wiedzy na temat pewnych rozwiązań. Dopiero w kolejnych etapach wytwarzania oprogramowania lub w momencie implementacji danego rozwiązania, można przekonać się, że nie jest ono poprawne.

Brak wiedzy o wadliwym kalendarzu

Na etapie projektowania założono, że da się łatwo wykonać integrację kalendarza należącego do zestawu narzędzi AdminLTE. Pracowano nad tym przez kilka tygodni, jednak okazało się, że ten komponent nie jest kompatybilny z naszym systemem. Gdybyśmy wcześniej wiedzieli o tym, że to rozwiązanie jest złe, zyskalibyśmy czas na inne rozwiązania implementacyjne.

Wybór optymalnego rozwiązania

Zarówno w trakcie fazy analizy jak i projektowania uznawano pewne pomysły za dobre i decydowano się je implementować. Jednak z czasem zachodzące zmiany w projekcie, jak i świadomość tego co wiąże się z wybranym rozwiązaniem, powodowały, że porzucano pewne pomysły lub zmieniano. Pojawia się tutaj trudność, ponieważ osoba, która mogła zacząć implementację, musi porzucić dotychczasowe prace i rozpocząć działanie nad nowym rozwiązaniem. Dodatkowo łączy się to z problemem organizacyjnym w postaci braku informowania pozostałych członków zespołu o postępach i napotkanych problemach, co opisane zostało w części pt. „Brak odpowiedniego przepływu informacji”.

Przykładem problemu, jaki się pojawił w projekcie, może być połączenie dotyczące matki i ojca danego konia. Początkowo zostało zastosowane rozwiązanie z referencjami, ale z czasem okazało się, że należy przechowywać te informacje jako String. Powodem był między innymi problem techniczny, bo Entity Framework nie pozwala na prostą implementację dwóch referencji w jednej klasie, która ma być mapowana. Dodatkowo informacja przechowywana

w zmiennej String okazała się być na potrzeby systemu wystarczająca. Zmiana nie była bardzo problematyczna czy skomplikowana, ale w przypadku, gdyby pozostałe części korzystały z referencji, byłby to problem.

Innym przykładem może być też próba zaimplementowania odpowiednio w modelu komentarzy dla Lekcji wystawianych przez Klienta. Na początku uznaliśmy, że właściwym rozwiązaniem będzie użycie klasy pośrednicząca *KlientLekcja* pomiędzy klasą *Lekcja*, a Użytkownikiem. Poprzez to połączenie mieliśmy wiedzieć kto i w jakiej lekcji dodał jakiś komentarz. Z czasem okazało się, że tak naprawdę część ta jest nieodpowiednim, niepotrzebnie skomplikowanym rozwiązaniem. Doszliśmy do takiego wniosku, bo musimy tworzyć dodatkowy obiekt pośredniczący pomiędzy Użytkownikiem, a Lekcją, który nie daje nam niczego dodatkowego, poza faktem, że musimy tworzyć oddzielny obiekt dla komentarza. Zastosowano zamiast tego atrybut typu String o nazwie Comments, a asocjacja do Użytkownika wskazywała, kim jest komentujący.

7.2.2 Trudności organizacyjne

W tym podrozdziale omówione zostały poszczególne problemy związane z organizacją projektu i pracą zespołową.

Zależność poszczególnych obiektów w systemie i terminy

W trakcie pisania rozbudowanego systemu często wydziela się pewne części, które nie powinny być powiązane ze sobą. W najlepszym wypadku dany programista pracuje na całkowicie oddzielonych od reszty zadań. Nie zawsze jednak tak jest. Poszczególne kawałki programu niekiedy są ze sobą bardzo mocno powiązane i bez poszczególnych części nie da się prowadzić dalszej pracy.

Problemem, jaki się u nas w związku z tym się pojawiał, był brak możliwości prowadzenia dalszych prac przez niektórych członków zespołu, ze względu na opóźnienia spowodowane niewykonaniem przez inną osobę potrzebnego komponentu na czas. Przykładem może być uzupełnianie bazy danych przykładowymi danymi, pozwalające na przyjrzenie się jak wyglądają wyświetlane obiekty. Dzięki temu można przeprowadzić odpowiednie zmiany w kodzie HTML, JavaScript czy CSS, w celu poprawienia czytelności strony. Niestety nie zawsze da się przewidzieć układ danych na stronie, więc brak możliwości wyświetlenia danych utrudnia dalsze prace programistyczne. Brak przykładowych danych

powodował, że osoba odpowiedzialna za implementacje widoku do wyświetlania wizyt weterynaryjnych, nie była w stanie dalej pracować nad programem.

Innym przykładem może być też brak napisanego API dla aplikacji mobilnej. Aby działać zgodnie z całym systemem i mieć spójne dane, aplikacja mobilna potrzebuje danych udostępnionych przez system w postaci API. Jego brak uniemożliwiał częściowo dalszą pracę nad aplikacją.

Nie zawsze przyczyną musi być niedotrzymanie danego terminu. Czasem złe oszacowanie długości tworzenia komponentu może prowadzić do tego, że jedna osoba nie może wykonać zleconego zadania na czas, co skutkuje wydłużonym czasem oczekiwania innych członków zespołu.

Brak odpowiedniego przepływu informacji

W projekcie grupowym istotne jest wzajemne informowanie się o wszelkich wydarzeniach, mogących mieć znaczny wpływ na postęp prac. Brak odpowiedniego przepływu informacji, m.in. o napotkanych problemach czy wykorzystanych rozwiązaniach, skutkuje opóźnieniem prac oraz powtarzaniem podobnych błędów przez innych członków grupy.

Podany przykład dotyczy fazy implementacji, niemniej jednak obrazuje również problemy organizacyjne. Kilku członków zespołu zajmującego się pisaniem programu, nie wiedziało o stosowaniu Html helpera o nazwie *EnumDropDownListFor*. Brak tej wiedzy powodował, że próbowali niepoprawnie lub niepotrzebnie implementować listę rozwijaną dla danych typu enum.

Innym przykładem jest też brak porozumienia co do tworzenia graficznej części strony. Początkowo każdy indywidualnie tworzył szablony CSS na potrzeby tworzonego przez siebie widoku, co zaburzało spójność i przejrzystość interfejsu użytkownika. Dopiero po pewnym czasie zostały stworzone uniwersalne szablony CSS wykorzystywane w każdym widoku.

7.2.3 Trudności techniczne

W projektach zdarzają się trudności związane z implementacją. Mogą one wynikać m.in. z braku wiedzy i umiejętności członków zespołu, nieodpowiednią wersją wykorzystywanych technologii czy niepoprawną pracą z repozytorium kodu.

Praca z systemem wersjonowania kodu github i technologią Pull Request

System wersjonowania kodu, jakim jest github, stanowi bardzo pomocne narzędzie do pracy z projektem. Istnieją różne strategie łączenia poszczególne części projektu w całość. Przykładowo wszyscy programiści mogą pracować na jednej gałęzi (ang. branch) albo robić tzw. prośbę o połączenie (ang. pull request), czyli próbować przyłączyć swoją gałąź do innej dostępnej. Pisząc nasz system skorzystaliśmy z drugiej strategii.

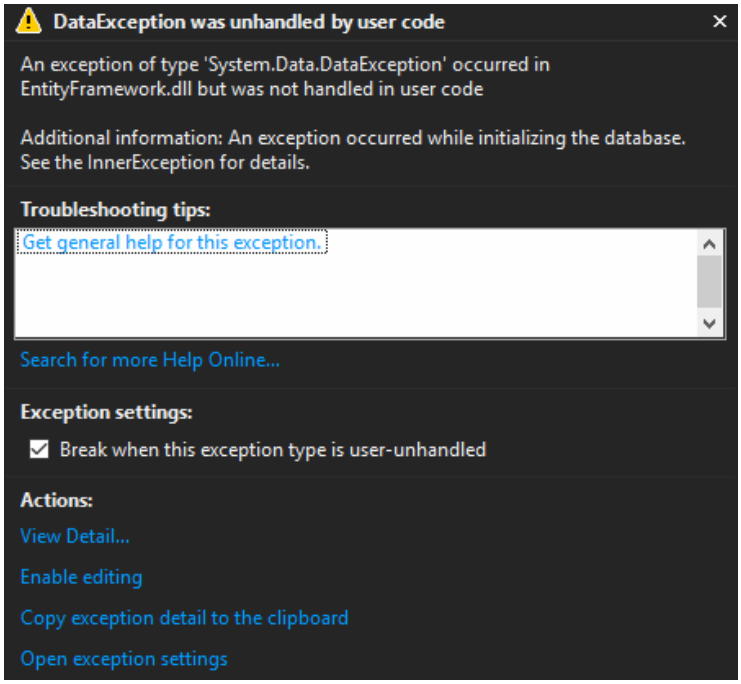
Zaletą takiego rozwiązania jest fakt, że wszelkie zmiany są stale monitorowane przez osobę akceptującą przyłączanie gałęzi do głównej części projektu, przez co niekorzystne lub złe rozwiązania mogą być niewciągnięte do projektu na etapie rozpatrywania prośby o połączenie. Ponadto osoba sprawdzająca nasz kod w trakcie przyłączania gałęzi jest w stanie spojrzeć na niego inaczej niż my po kilku godzinach pracy z nim i szybko zobaczyć jakieś błędy, których my po długim czasie pracy z programem, możemy nie zauważać. Dlatego gdyby doszło do jakiejś pomyłki w trakcie tworzenia oprogramowania, można ją wykryć w trakcie realizowania prośby o przyłączenie, a więc przed dodaniem czegoś do części produkcyjnej.

Niestety przed skorzystaniem z czyjegoś rozwiązania, trzeba poczekać aż zostanie ono wciągnięte do gałęzi produkcyjnej.

Brak wystarczających informacji o błędzie

W trakcie samego pisania niekiedy dochodzi do błędów, które pokazywane są w postaci wyjątków. Wyjątki rzucane są np. gdy wychodzimy poza zakres tablicy, kiedy niepoprawnie przypisujemy jakiś obiekt lub gdy po prostu dokonujemy jakiejś niewłaściwej operacji, wykrywanej dopiero na etapie działania programu (ang. Run Time). Problematyczna staje się sytuacja, w której, pomimo informacji o błędzie, nadal nie wiemy co zrobić, bo komunikat nie zawiera istotnych dla nas szczegółów.

Przykładem takiej sytuacji w naszym systemie jest działanie Fluent API i DataAnnotations na modelach. Przy pomocy tych dwóch technologii można nanosić ograniczenia na dane modele np. ustalić, że dana wartość musi zostać utworzona z jakąś wartością początkową inną niż null. Problem pojawiał się w momencie, w którym ktoś próbował stworzyć obiekt bez uprzedniego zainicjalizowania, nie wiedząc o ograniczeniu. W takim przypadku pojawia się wyjątek typu System.Data.DataException. Nie wiadomo do końca co go spowodowało i jak go rozwiązać. Informacja o wywołaniu wyjątku pokazana jest na Rysunek 76.



Rysunek 76. Błąd wywołany przez źle zainicjalizowany obiekt

Dodatkowym problemem pojawiającym się tutaj jest trudność organizacyjna. Jeśli osoba odpowiedzialna za fluent API lub DataAnnotations nie powiadomi innych programistów o tym, że utworzyła jakieś adnotacje, pozostali członkowie grupy nie będą wiedzieć, co może być przyczyną zaistniałego błędu.

Zmiany globalnie wpływające na system

Przy tworzeniu projektu często ma się dostęp do kodu, który wpływa globalnie na cały system. Mogą to być zarówno zmienne jak i metody. Ważną ich cechą jest to, że są powiązane z wieloma składowymi programu. Zmiana takiej części może być problematyczna, ponieważ istnieje ryzyko, że korzystające z niej obiekty przestają działać.

Przykładem takiej globalnie działającej części jest metoda *SaveChangesAsync()*, z której korzysta znacząca część akcji w kontrolerach. W przypadku próby zmiany takiej składowej programu, może dojść do błędów rozchodzących się na każdy obiekt, który z niej korzysta. Jeśli potrzebujemy tej zmiany do własnej części programu, przed naniesieniem zmian musimy upewnić się, że nie wpłyną one negatywnie na pozostałe obiekty lub nie zmieni się ich dotychczasowe działanie.

Inną trudnością tego typu, jaka się pojawiała w naszym systemie, była zmiana modeli. Usunięcie zmiennej wykorzystywanej w innych częściach kodu prowadziło do kaskadowych zmian, które były zazwyczaj niekorzystne.

Kompilacja plików .cshtml w Visual Studio 2015

Visual Studio 2015 ma możliwość kompilowania kodu HTML, co jest niezwykle pomocne w pracy z front-end'em. Dzięki temu łatwo możemy dowiedzieć się czy nie popełniamy błędu. Przykładowo jesteśmy informowani o tym, że tag jest niedomknięty.

Niestety system sprawdzający składnię plików .cshtml ma też wady. Po otworzeniu pliku bardzo często większość linii kodu podświetlona jest na czerwono i raportowane są błędy. Sytuacja ta pojawia się niekiedy nawet na kilka sekund. Powodowało to niekiedy podejrzenie, że napisany przez nas wcześniej program jest błędny, choć w rzeczywistości taki nie był.

Ponadto IntelliSense dla plików .cshtml nie działa tak dobrze, jak moglibyśmy tego oczekiwać. Przykładem może być „znikająca" lista rozwijana, która ma pokazywać jakich pól danego obiektu możemy użyć. Lista pojawia się na krótką chwilę, a jeśli nie użyjemy skrótu Ctrl+Enter, podpowiedź zniknie zanim zdążymy wybrać którąkolwiek z opcji.

7.3 Proponowane kierunki rozwoju

Brak doświadczenia, czasu, możliwości rozmów z pracownikami stadnin oraz ograniczone zasoby spowodowały, że na chwilę obecną system nie odpowiada na wszystkie potrzeby osób zarządzających ośrodkami jeździeckimi. W trakcie tworzenia systemu zidentyfikowano możliwe kierunki rozwoju systemu, które uczynią go nie tylko kompletnym narzędziem do zarządzania stadniną, ale także ułatwią klientom i pracownikom funkcjonowanie w środowisku stajennym.

Zarządzanie finansami

Pierwszym elementem, o który należałoby rozszerzyć aplikację SIDOS jest zarządzanie finansami. Taka funkcjonalność nie została zaimplementowana ze względu na zbyt dużą złożoność. Realizacja wymagałaby konsultacji z osobami zajmującymi się finansami na co dzień. Zarządzanie płatnościami obejmowałoby takie funkcje, jak:

- Wystawianie faktur;
- Śledzenie wpływów i wydatków;
- Wykonywanie raportów finansowych oraz zestawień utargów.

Pracownik księgowy stajni miałby możliwość ręcznego wprowadzania wydatków oraz przychodów do systemu. Z tak zapisanych danych możliwe byłoby przygotowanie zestawienia

wpływów i wydatków ośrodka. Taka funkcjonalność znacząco ułatwiłaby śledzenie finansów ośrodka.

Część systemu odpowiadająca za zarządzanie finansami mogłaby również zostać rozszerzona o możliwość internetowej płatności za jazdy za pomocą serwisu, np. t-pay. Taka opcja zmniejszyłaby ryzyko pomyłki podczas podliczania gotówkowych wpłat za usługi.

Forum

W celu usprawnienia komunikacji między klientami kolejnym możliwym elementem rozszerzającym system jest forum dostępne dla zalogowanych użytkowników. Byłaby to funkcjonalność stworzona z myślą o klientach i właścicielach koni ułatwiająca rozwiązywanie prywatnych problemów związanych z jeździectwem czy opieką nad końmi. Obecność na forum instruktorów i trenerów zatrudnionych przez stajnię umożliwia udzielenie odpowiedzi przez profesjonalistę i minimalizuje ryzyko otrzymania odpowiedzi niezgodnej z prawdą. Duża liczba użytkowników umożliwiałaby zdobywanie wiedzy z wielu źródeł.

Ogłoszenia o rekrutacji

Aby ułatwić pozyskiwanie nowych pracowników do strony powitalnej zostałaby dodana sekcja "rekrutacja". Aktualnie oferty pracy dodawane są jako aktualności, jednak wydzielenie sekcji rekrutacji pozwalałoby odwiedzającym stronę zapoznanie się z aktualnymi ofertami, a po zalogowaniu do systemu z ich szczegółami. Dodatkowo takie rozwiązanie ułatwia nawiązanie współpracy z ośrodkiem osobom już w pewien sposób z nim związanym, na przykład jego klientom. Osoba regularnie logująca się do systemu będzie mogła szybko zorientować się, czy ośrodek aktualnie rekrutuje nowych pracowników. Zwiększa to szanse zatrudnienia osoby znanej i zaufanej, co może pozytywnie wpłynąć na odbiór ośrodka przez klientów i panującą w nim atmosferę.

Oferty sprzedaży

Ponadto system może zostać rozszerzony o sekcję sprzedaży. W tym miejscu właściciele koni, klienci i pracownicy mogliby dodawać oferty sprzedaży czy kupna koni prywatnych. Osoby związane z ośrodkiem, zainteresowane kupnem konia, miałyby możliwość w pierwszej kolejności przejrzeć oferty stajenne. Dopiero w przypadku nieznalezienia interesującej oferty kontynuowaliby szukanie w innych miejscach. Takie rozwiązanie oszczędza dużo czasu zarówno kupującemu, jak i sprzedającemu.

Tagi

Elementem ułatwiającym klientom korzystanie z systemu byłyby tagi. Aktualnie SIDOS wykorzystuje tagowanie tylko do charakteru koni, jednak funkcjonalność ta mogłaby zostać rozszerzona na inne części systemu. Tagi w aktualnościach ułatwiałyby wyszukiwanie tych wpisów, które interesują klienta, na przykład "zawody", "warsztaty" czy "wykłady". Tagowanie sprzedaży pomogłoby filtrować oferty na przykład po charakterze konia, jego przeznaczeniu (np. rekreacja, sport, dyscyplina) czy wymaganym poziomie zaawansowania potencjalnego przyszłego właściciela. System tagów znacznie przyspiesza i ułatwia wyszukiwanie informacji, które interesują użytkownika.

Rozszerzenie systemu o powyższe elementy nie tylko uczyniłoby go kompletnym narzędziem wspomagającym zarządzanie ośrodkiem jeździeckim, ale również sprawiło, że byłby on bardziej przyjazny użytkownikom. Sprzedaże, oferty pracy czy tagi pozwalają na zaoszczędzenie czasu i ułatwiają korzystanie z systemu poprzez uporządkowanie informacji, natomiast część finansowa umożliwia przechowywanie wszystkich elementów dotyczących finansów ośrodka w jednym miejscu, bez konieczności przełączania się między systemami (np. SIDOS a systemem finansowym).

8 Bibliografia

[1] https://mikroporady.pl/pomocne-definicje/43-dzierzawa.html, data dostępu: 06.01.2018

[2] https://mattermark.com/companies/aluasoft.com, data dostępu: 13.01.2018

[3] https://www.capterra.com/p/118836/Paddock-Pro/, data dostępu: 16.12.2017

[4] www.paddockpro.com/products/ppro_tour/index.html, data dostępu: 16.12.2017

[5] https://www.capterra.com/p/146345/HorseCount/, data dostępu: 19.12.2017

[6] https://www.horsecount.com/, data dostępu: 19.12.2017

[7] http://www.barnmanager.net/, data dostępu: 31.01.2017

[8] https://www.capterra.comp/164823/BarnManager/, data dostępu 21.12.2017

[9] https://www.barnmanager.com/, data dostępu: 21.12.2017

[10] http://www.computerexperts.pl/p/net-framework-w-skrocie-net,57, data dostępu: 06.01.2018

[11] https://www.ecma-international.org/publications/standards/Ecma-335.htm, Standard ECMA-335, data dostępu: 06.01.2018

[12] https://martinfowler.com/eaaCatalog/repository.html, data dostępu: 28.05.2017

[13] https://en.wikipedia.org/wiki/Microsoft_Visual_Studio, data dostępu: 21.11.2017

[14] https://msdn.microsoft.com/pl-pl/library/bb386063.aspx, data dostępu: 21.11.2017

[15] https://en.wikipedia.org/wiki/Gradle, data dostępu: 10.01.2018

[16] https://en.wikipedia.org/wiki/Lint_(software), data dostępu: 10.01.2018

[17] https://www.holaxprogramming.com/2017/07/04/devops-gradle-is-faster-than-maven/, data dostępu: 28.10.2017

[18] ECMA-334 5th Edition / December 2017, https://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf, str. xx, data dostępu: 06.01.2018

[19] https://stackoverflow.com/questions/4233112/what-is-a-unified-type-system, data dostępu: 11.02.2018

[20] Joseph Albahari, Ben Albahari, „C# 6.0 Leksykon kieszonkowy”, ISBN: 978-83-283-2446-6

[21] https://docs.microsoft.com/pl-pl/dotnet/csharp/language-reference/keywords/index, data dostępu: 24.11.2017

[22] https://www.nuget.org/packages/Microsoft.AspNet.Mvc, data dostępu: 29.01.2018

[23] https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/adding-a-controller, data dostępu: 29.01.2018

[24] https://www.w3schools.com/jquery/jquery_traversing.asp, data dostępu: 16.11.2017

[25] https://www.w3schools.com/xml/ajax_xmlhttprequest_create.asp, data dostępu: 21.11.2017

[26] Chris Ullman, Lucinda Dykes, „Ajax od podstaw”, Gliwice, Helion, 2008, ISBN 978-83-246-1287-1

[27] https://en.wikipedia.org/wiki/Ajax_(programming), data dostępu: 21.11.2017

[28] http://blog.debugme.eu/best-alternatives-to-javascript/, data dostępu: 10.01.2018

[29] https://forum.pasja-informatyki.pl/questions/programowanie/javascript-jquery-ajax, data dostępu: 10.01.2018

[30] https://en.wikipedia.org/wiki/Netscape, data dostępu: 10.01.2018

[31] https://en.wikipedia.org/wiki/TypeScript, data dostępu: 10.01.2018

[32] https://www.w3schools.com/js/, data dostępu: 10.01.2018

[33] https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282, data dostępu: 10.01.2018

[34] https://en.wikipedia.org/wiki/Entity_Framework, data dostępu: 25.11.2017

[35] Mariusz Trzaska, “Data Migration and Validation Using the Smart Persistence Layer 2.0”. The 16th IASTED International Conference on Software Engineering and Applications (SEA 2012). Editors: T. Gonzalez, M.H. Hamza. Acta Press. ISBN: 978-0-88986-951-6. November 12 - 14, 2012. Las Vegas, USA. pp. 186-193

[36] https://en.wikipedia.org/wiki/Dependency_injection, data dostępu: 02.12.2017

[37] https://msdn.microsoft.com/en-us/library/ff649690.aspx, data dostępu: 02.12.2017

[38] http://www.testowanie.net/poziomy-testow/testy-systemowe/, data dostępu: 19.01.2018

[39] http://www.testowanie.net/testowanie/poziomy-i-typy-testow/, data dostępu: 19.01.2018

[40] http://www.testowanie.net/poziomy-testow/testy-modulowe-unit-tests/, data dostępu: 19.01.2018

[41] https://dariuszwozniak.net/2013/05/28/kurs-tdd-czesc-2-testy-jednostkowe-a-testy integracyjne/, data dostępu: 19.01.2018

[42] http://www.testerzy.pl/artykuly/poziomy-testowania, data dostępu: 19.01.2018

[43] http://www.testowanie.net/poziomy-testow/testy-systemowe/, data dostępu: 19.01.2018

Załącznik A. Organizacja pracy grupy

Pisanie naszej pracy rozpoczęliśmy od zaplanowania faz: analizy, projektowania, implementacji, dokumentacji i testów. Dla każdej fazy został stworzony zespół, który odpowiedzialny był za realizację założeń.

Lider grupy: Magdalena Popek

Zespół analizy:
Lider: Paulina Gruba
Członkowie:

1. Paulina Stasiów
2. Dominik Deja
3. Szymon Ritz
4. Piotr Prystupa
5. Patryk Pohnke

Zespół projektowania:
Lider: Jacek Muszyński
Członkowie:

1. Paulina Stasiów
2. Michał Skowroński
3. Piotr Prystupa
4. Dominik Deja

Zespół implementacji:
Lider: Piotr Prystupa
Członkowie:
Aplikacja webowa:

1. Paulina Stasiów
2. Szymon Ritz
3. Patryk Pohnke
4. Jacek Muszyński

Aplikacja mobilna:

1. Dominik Deja

Zespół dokumentacji:

Lider: Paulina Gruba

Członkowie:

1. Jacek Muszyński
2. Paweł Walesiak
3. Michał Skowroński
4. Patryk Pohnke

Zespół testowy:

Lider: Magdalena Popek

Członkowie:

1. Paulina Gruba
2. Jacek Muszyńki
3. Paweł Walesiak

Faza analizy:

Faza analizy rozpoczęła się 23 stycznia 2017 i zakończyła się 31 marca 2017, miała za zadanie określić jakie funkcje na spełniać system, dla jakiego grona odbiorców powinien zostać dostosowany. Pierwszym etapem fazy analizy była tzw. „burza mózgów" – każda osoba z zespołu mówiła, jaki według niej powinien być system. Zgłoszenia dotyczyły:

- Funkcjonalności, jakie powinien spełniać system
- Ról i użytkowników, dla których powinien być dostępny i dostosowany
- Jakie działania powinien umożliwiać osobom z niego korzystających
- Jak, z punktu widzenia użytkownika, powinny wyglądać niektóre funkcjonalności (szkic interfejsu)
- Czego w systemie być nie powinno, jakie opcje powinny być zablokowane dla użytkowników

Następnie, założenia te zostały zebrane i na ich podstawie powstały

- Diagram klas
- Diagramy przypadków użycia
- Diagramy aktywności

Na zakończenie fazy analizy został utworzony dokument fazy analizy podsumowujący postanowienia
i założenia, jakie powinny zostać zrealizowane.

Faza projektowa

Faza projektowania rozpoczęła się 1 kwietnia 2017, a zakończyła się 29 czerwca 2017. Zadaniem fazy projektowania było przeanalizowanie i modyfikacja diagramów powstałych podczas fazy analizy uwzględniając szczegóły implementacyjne.
Wynikiem fazy projektowania są uaktualnione diagramy i dokument fazy projektowania.

Faza implementacji

Faza implementacji rozpoczęła się 30 czerwca 2017 i trwała do 15 stycznia 2018. Podczas fazy implementacji grupa implementująca aplikację webową spotykała się osiem razy. Podczas tych spotkań członkowie grupy kontynuowali pracę nad swoimi zadaniami, omawiali napotkane trudności i razem szukali rozwiązań. Spotkania grupy implementacji były okazją do konsultacji rozwiązań i ewentualnie ich modyfikację na podstawie wniosków.

Równolegle do prac nad aplikacją webową trwały prace nad aplikacją mobilną, której implementacją zajmował się Dominik Deja.

Wynikiem fazy implementacji są aplikacje mobilna i webowa oraz dokument fazy implementacji, opisujący obie aplikacje.

Faza dokumentacji

Grupa dokumentacji rozpoczęła pracę 4 lipca 2017, zakończyła 31 stycznia 2018. Do zadań grupy należało sporządzenie dokumentów poszczególnych faz oraz napisanie tekstu pracy inżynierskiej.

Pierwsze zadania zostały przypisane członkom grupy na podstawie konspektu – każda osoba dostała do napisania poszczególne podrozdziały pracy. W związku z byciem przydzielonymi do grupy implementacji i dokumentacji Jackowi Muszyńskiemu oraz Patrykowi Pohnke zostało przydzielone mniej prac do napisania ze względu na konieczność wywiązywania się z zadań przydzielanych im w grupie implementacji.
Każda napisana praca przechodziła kilka etapów. Podstawowy obieg dokumentów:

1. Napisanie przez autora

2. Sprawdzenie przez lidera grupy – naniesienie komentarzy
3. Edycja pracy - uwzględnienie uwag z 2. przez autora
5. Cross chceck – sprawdzenie pracy przez inną osobę z grupy
5. Final check – sprawdzenie pracy przez Magdalenę Popek; wynikiem kroku była akceptacja lub powrót do punktu 3.

Niektóre prace przechodziły więcej niż jedną iterację, tj. były poprawiane przez autora więcej niż podstawowe dwa razy.

Wynikiem fazy dokumentacji jest tekst pracy inżynierskiej.

Faza testów

Faza testów rozpoczęła się 8 stycznia 2018 a zakończyła 28 stycznia 2018. Do zadań grupy należało sprawdzanie zgodności systemu z założeniami, sprawdzenie poprawności działania poszczególnych funkcjonalności oraz funkcjonalności zależnych od siebie. W przypadku stwierdzenia błędu funkcjonalności tester informował o nim grupę implementacji poprzez wystawienie zgłoszenia na platformie GitHub w sekcji Issues. Po otrzymaniu informacji o naprawie błędu funkcjonalność była ponownie poddana testom w celu sprawdzenia poprawności działania. W przypadku pozytywnego wyniku testu osoba testująca informowała o tym grupę implementacji i zamykała wystawione zgłoszenie. W przypadku wyniku negatywnego do zgłoszenia dołączano załącznik (najczęściej zrzut ekranu) z błędem, w celu ułatwienia grupie implementacji poprawnego wykonania zleconego zadania.

Załącznik B. Lista zmian dokonanych w trakcie fazy projektowania

Faza projektowania ma na celu udoskonalenie diagramów stworzonych w fazie analizy oraz rozwinięcie wątków dotyczących wyglądu i sposobu implementacji aplikacji SIDOS. W zespole pięcioosobowym, którego liderem był Jacek Muszyński dokonano zmian i poprawek diagramów rozpoczętych w poprzedniej fazie analizy.

10 Marca 2017

Zmiany dokonane w projektowym diagramie klas:

Zaimplementowano kompozycje, usunięto klasę „Wniosek o przedłużenie dzierżawy".

Dokonano zmian w klasie Użytkownik decydując, że zdjęcia użytkownika będzie wgrywane na stronę. Uznano też, że historia wizyt weterynaryjnych będzie do przeglądu dla użytkownika.

W klasie Wiadomość dodano możliwość załączania plików (głównie chodzi tu o zdjęcia).

W klasie Zadanie zmieniono typy zmiennych, żeby odpowiadały typom używanym przez C#.

W klasie Karmienie stworzono klasę Pasza, która zawierała zmienne takie jak „waga", „składniki".

W klasie podawanie leków stworzono klasę Lek zawierającą np. składniki, wagę i usunięto listę pasz.

W klasie Pracownik zamiast metody edytuj galerię dodano metody dodaj zdjęcie, edytuj zdjęcie. Dodano oznaczanie zadań jako wykonanych, przeglądanie swoich zadań, akceptowanie zadań, zmianę danych kontaktowych, akceptowanie aktualizacji zadania i ich odrzucanie.

Stworzono klasę zdjęcie zawierającą takie zmienne jak adres pliku i jego nazwa.

W klasie klient dodano następujące funkcjonalności: umów się na lekcję, odwołaj lekcję, wyświetl zaplanowane lekcje.

12 Marca 2017

W klasie kierownik dodano lub zmieniono następujące funkcje:

DodajKonia() jako argumenty przyjmujące argumenty klasy Koń,

DodajPracownika jako argumenty przyjmująca argumenty klasy Koń,

PrzeglądajWszystkieZadania() bez argumentów

PrzypiszDzierżaweDoKonia() od teraz przyjmuje jako argument obiekt klasy Koń

RozpatrzenieWnioskuODzierżawę od teraz przyjmuje jako argument obiekt klasy Dzierżawa
i zwraca boolean

UsuńKonia() od teraz przyjmuje jako argument obiekt klasy Koń i zwraca boolean.

Ponadto w klasie DzierżawaKonia usunięto pamiętanie ceny w przypadku niepełnego miesiąca.

13 Marca 2017

Certyfikat od teraz będzie klasą z atrybutami tytuł, opis, data.

Zdecydowano się na nieusuwanie funkcji aktualizacjaZadania().

Usunięto klasy Pasza i Lek, zastąpiono je klasą Karmienie, zawierającą takie argumenty jak: rodzaj karmienia, dawka, nazwa.

Przesunięto metodę zmienHaslo() do Użytkownika.

Usunięcie wszystkich metod usuwających. Tą decyzję uargumentowano potrzebą przechowywania wszystkich informacji w systemie. Zamiast tego np. metoda ukryjKonia()

W klasie Kierownik zmieniono metodę zmienGrafikeHali() na zmienGrafikLokalizacji().

W klasie Lokalizacja dodano zmienną będącą obrazkiem.

Zastąpiono klasę Zdjęcie kolekcją Stringów zawierających URL

W klasie WizytaWeterynaryjna wyrzucono metodę dodajWpis()

Usunięto klasę WniosekODzierżawę, zastąpiono je klasą DzierżawaKonia

Zmiany wprowadzone po marcu 2017

Dokonano spłaszczenia dziedziczenia w przypadku klasy Pracownik, dodając mu atrybut rodzajPracownika(enum). Klasa PRACOWNIK ma teraz wszystkie atrybuty i metody klas dziedziczących po niej, a owe klasy usunięto z diagramu. Użycie podejścia Table Per Hierarchy w tym przypadku wydaje się być najlepszym rozwiązaniem z uwagi na znaczne uproszczenie diagramu klas, a przez to, znaczne uproszczenie implementacji. Dodano też dziedziczenie typu overlapping, aby umożliwić instruktorowi bycie również klientem, czy stajennemu bycie weterynarzem itp.

Pozostałe zmiany:

1. Usunięto klasę OsobaDzierżawiąca oraz Wniosek o Dzierżawę.
2. Dokonano spłaszczenia dziedziczenia w przypadku klasy Zadanie.
3. Usunięto klasę Dzierżawa oraz Wniosek o Dzierżawę i zastąpiono je klasą Dzierżawa Konia, która przyjmuje wszystkie asocjacje klasy Wniosek o Dzierżawę, ale ma jednocześnie pola i metody klasy Dzierżawa.
4. Usunięto klasę Zdjęcie.
5. W klasie Koń dodano listę cech charakteru. Atrybut asocjacji między klasą Koń, a Właściciel został zamieniony na klasę pośredniczącą *KońWłaściciel*.
6. Atrybut asocjacji klas Klient i Lekcja zmieniony został w klasę asocjacji (klasę pośredniczącą) *KlientLekcja*.

9 Spis ilustracji

10 Listingi