Studia i Materiały Informatyki Stosowanej, Tom 12, Nr 1, 2020
str. 5-9

Aplikacja na platformę Android z systemem ekspertowym jako rozwiązanie problemu właściwego żywienia koni

Adrianna Piszcz[1] **Dariusz Mikołajewski**[2]

[1] *Studentka 1 rok MU Informatyka, Instytut Informatyki, Uniwersytet Kazimierza Wielkiego w Bydgoszcz*
[2] *Uniwersytet Kazimierza Wielkiego, Instytut Informatyki, Kopernika 1, 85-074 Bydgoszcz*

Streszczenie: *Celem badania jest stworzenie aplikacji wspomagającej prawidłowe żywienie koni na platformę Android. Aplikacja realizowana jest w środowisku Xamarin w programie Visual Studio z wykorzystaniem języka C#. Dane niezbędne do działania aplikacji magazynowane są w plikach XML. Model systemu ekspertowego został przygotowany w oparciu o drzewa decyzyjne, natomiast projekt aplikacji opisuje diagram przypadków użycia oraz diagram klas. System ekspertowy, który został zaimplementowany w aplikacji, wykorzystuje do wygenerowania diety bazę wiedzy pozyskaną od ekspertów oraz moduł wnioskujący. Aplikacja może służyć zarówno celom użytkowym, jak i edukacyjnym. Będzie użyteczna dla osób ze specjalistycznym doświadczeniem, jak i bez niego.*

Słowa kluczowe: *Xamarin, aplikacja Android, XML, żywienie koni, system ekspertowy*

Application for the Android platform with the system as a solution to the right problem horse nutrition

Abstract: *The aim of the study is to create application supporting the proper nutrition horses on the Android platform. The application is created in the environment Xamarin in Visual Studio IDE using CSharp language. The data needed for the application are stored in XML files. Expert system model was developed based on decision trees and the design of the application describes the use case diagram and class diagram. Expert system used in the application generates a diet using a knowledge base gained from experts and requesting module. The application can be used both practical purposes, as well as educational. It will be useful for people with specialized experience, and without it.*

Keywords: *Xamarin, Android application, XML, horse nutrition, expert system*

1. Wprowadzenie

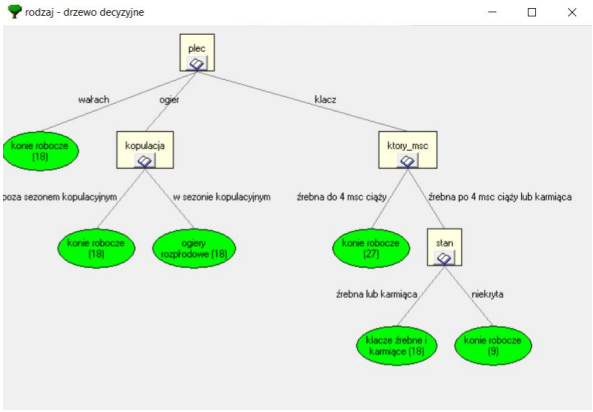
Jeździectwo nieustannie się rozwija, stajnie stają się coraz bardziej rozbudowane i innowacyjne. Wzrasta świadomość i wiedza właścicieli oraz hodowców koni. Jednak nie sposób wymagać od wszystkich takiej samej wiedzy na temat żywienia, a jest to jeden z głównych aspektów zapewniających dobrobyt zwierzęcia. Często w stajniach wszystkie konie skarmiane są tymi samymi dawkami pokarmowymi, niezależnie od ilości ruchu, rasy czy innych czynników determinujących dobór optymalnej diety.

System ekspercki opiera się na wiedzy, która dostarcza rozwiązania problemów dotyczących danego zagadnienia poprzez wnioskowanie na podstawie bazy wiedzy. Bazy wiedzy tworzone są na podstawie umiejętności ekspertów, ich wiedzy merytorycznej, toku rozumowania oraz decyzyjności.

Aplikacja ma rozwiązywać problemy żywienia koni w stajniach. System ekspertowy służyć będzie dobraniu najlepszej diety, poprzez zadawanie użytkownikowi szeregu pytań dotyczących konkretnego konia. Informacje te będą katalogowane w osobnej zakładce, aby użytkownik miał łatwy dostęp do każdej diety.

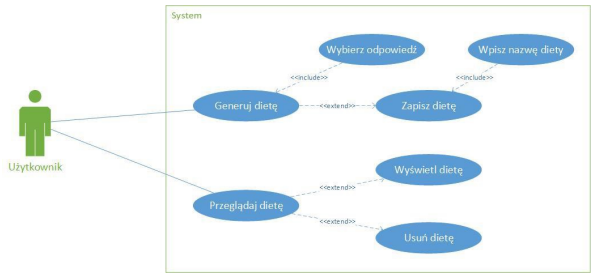
Celem jest zbudowanie intuicyjnego systemu, który dotrze do szerokiego grona odbiorców, niezależnie od stopnia wiedzy. System ten ma wspierać dobre praktyki żywieniowe koni. W łatwy sposób pozwoli dobrać oraz zapisać dietę dostosowaną dla konkretnego konia.

2. Metody



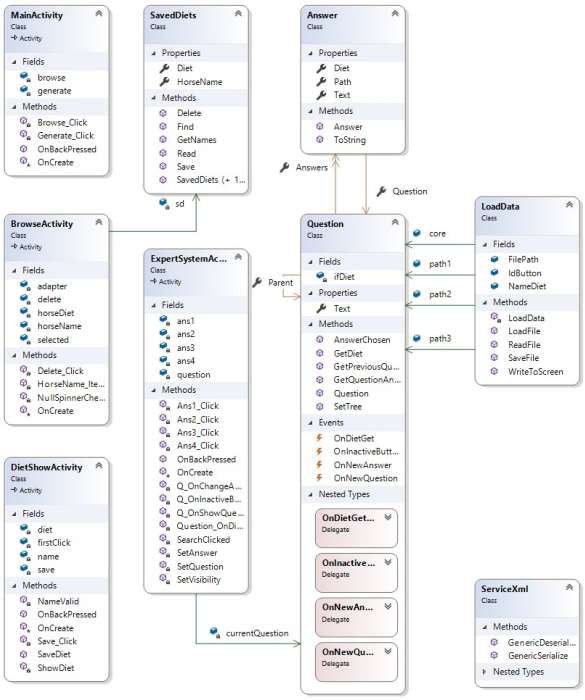
Rysunek 1: Drzewo decyzyjne grupy głównej - DETREEX

Drzewa decyzyjne Drzewo decyzyjne (Rys. 1) składa się z korzenia, krawędzi, węzłów oraz liści. Liście to węzły, z których nie odchodzą już żadne krawędzie. Krawędzie są połączeniami istniejącymi w modelu. Drzewa decyzyjne mają strukturę hierarchiczną. W kolejnych krokach dzielony jest zbiór obiektów, poprzez odpowiadanie na pytania o wartości wybranych atrybutów. Ostateczna decyzja zależy od odpowiedzi na wszystkie pytania. Projekt drzew decyzyjnych został wykonany za pomocą systemu DETREEX.



Rysunek 2: Diagram przypadków użycia

Diagram przypadków użycia Diagram przypadków użycia pozwala spojrzeć na system od strony użytkownika. Charakterystyczną grafiką figurki człowieka oznaczany jest aktor, czyli pewien określony rodzaj użytkownika. W diagramie może występować jeden lub kilku aktorów. Aktor określa użytkownika o sprecyzowanych uprawnieniach i o możliwych do podjęcia przez niego akcji. Natomiast elipsa reprezentuje przypadek użycia, to znaczy jakąś określoną akcję. W tym diagramie zawsze aktor inicjuje akcje. W realizowanej aplikacji występuje jeden typ użytkownika. Może on generować lub przeglądać dietę. Generowanie diety jest bezpośrednio związane z wybieraniem odpowiedzi. Dietę wygenerowaną można zapisać, a to z kolei wiąże się z poprawnym wpisaniem nazwy diety. Przeglądanie diet rozszerzone jest o ich usuwanie oraz o wyświetlenie składu wybranej diety. Wykonany w programie MS Visio diagram przypadków użycia (Rys. 2) obrazuje możliwe akcje wykonywane przez użytkownika systemu Paszomat.



Rysunek 3: Diagram klas

Diagram klas Klasa przedstawiona jest w postaci encji zawierającej atrybuty i funkcje, czyli działania jakie może wykonać obiekt tej klasy. Linie łączące poszczególne encje wskazują na relacje występujące pomiędzy klasami. Klasy mogą dziedziczyć z innych klas lub mogą być powiązane poprzez utworzenie obiektu jednej klasy w innej klasie. Na rysunku (Rys. 3) przedstawiono diagram klas systemu Paszomat. Po prawej stronie widoczne są dwie klasy serwisowe: klasa ServiceXML, która nie posiada żadnych powiązań, ponieważ jej metody są statyczne oraz klasa LoadData, która inicjalizuje obiekty klasy Question. W trzeciej kolumnie widoczne są klasy Question i Answer, które mają pomiędzy sobą połączenie obustronne. Klasa Question implementuje kolekcje obiektów Answer, a klasa Answer obiekt Question. Dodatkowo można zauważyć, że klasa Question posiada połączenie z samą sobą. Oznacza to, że wewnątrz tej klasy tworzona jest

jej instancja. Klasy aktywności nie posiadają powiązań z resztą systemu, ponieważ przepływ informacji pomiędzy nimi odbywa się za pośrednictwem eventów, za wyjątkiem klasy ExpertSystemActivity, która posiada powiązanie z klasą Question, aby pobierać informacje o aktualnym pytaniu oraz klasą BrowseActivity, która posiada powiązanie z klasą SavedDiets (implementuje instancję tej klasy), po to aby móc zarządzać zapisanymi danymi.

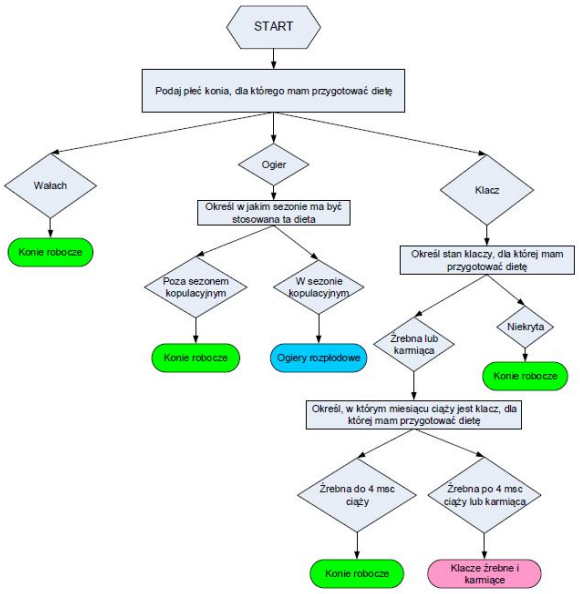
Baza danych XML Extensible Markup Language, w skrócie XML to język znacznikowy należący do tej samej rodziny języków co HTML. Odróżnia go fakt, że znaczniki nie są definiowane odgórnie poprzez język, tylko jest to rolą użytkownika. Zaletą przechowywania danych w plikach xml w porównaniu do tradycyjnej bazy danych jest prostota projektowania, możliwość szybkiego przeszukiwania i pozyskiwania danych. Baza danych XML została wykorzystana w projekcie do przechowywania bazy wiedzy niezbędnej dla funkcjonowania systemu ekspertowego. Jej struktura jest zgodna ze strukturą wymodelowaną na drzewach decyzyjnych. Projekt wykorzystuje znaczniki o nazwach identycznych jak nazwy klas oraz pól tworzących strukturę projektu.

Technologia Xamarin Xamarin pozwala na natywne projektowanie aplikacji mobilnych, co sprawdza się przy realizacji projektów nastawionych na multiplatformowość. Kod Xamarin.Forms pisany jest w XAML-u. Został przygotowany zestaw kontrolek, które mapowane są na kontrolki używane w danym systemie.

3. Rezultaty

System ekspertowy Projekt systemu doradczego, wspomagającego dobór optymalnej diety dla konia, składa się z części głównej, którą jest wybór grupy, do której należy koń, a także z trzech części podrzędnych stanowiących kontynuację dla wyboru z grupy głównej. Zatem powstały cztery odrębne modele, z których jeden, jest modelem głównym (Rys. 4) i determinuje przejście do jednego z trzech pozostałych.

Bazy wiedzy Na podstawie opracowanych diagramów utworzono odpowiednie pliki formatu XML, wykorzystane w późniejszym etapie do implementacji logiki systemu. Każdemu modelowi odpowiada jeden plik XML, tak więc analogicznie istnieje plik grupy głównej oraz trzy pliki grup szczegółowych. Każdy z plików posiada jednakową strukturę odpowiadającą strukturze klasowej projektu. Elementem głównym plików jest etykieta *Question*, która obejmuje wszystkie pozostałe



Rysunek 4: Diagram grupy głównej

elementy danego pliku i jest korzeniem dla całego budowanego systemu. Posiada dwóch bezpośrednich potomków: *Text* oraz *Answers*. Etykieta *Text* przechowuje treść pytania, natomiast *Answers* jest węzłem dla odpowiedzi na to pytanie. Każda odpowiedź ma treść i może prowadzić do jednego z trzech elementów: do kolejnego pytania, do odpowiedzi jaką jest dieta lub do ścieżki wskazującej na jeden z trzech plików szczegółowych.

Interfejs użytkownika Interfejs użytkownika został opracowany z bardzo dużą starannością. Aplikacja z założenia ma być prosta i przyjazna mniej technologicznym użytkownikom. Składa się ona z czterech różnych ekranów – menu głównego (Rys. 5), systemu ekspertowego, wygenerowanej diety oraz katalogu zapisanych diet. Pierwszym ekranem widocznym po uruchomieniu aplikacji jest menu główne. Dla wszystkich ekranów aplikacji został wykorzystany układ *RelativeLayout*, który charakteryzują zależności od innych elementów tego układu. Wybranie opcji „Generuj" przez użytkownika powoduje przekierowanie do najważniejszej części aplikacji – szeregu pytań i odpowiedzi prowadzących do jak najlepiej dobranej diety. Układ tego ekranu stanowi *TextView* wyświetlający treść aktualnego pytania oraz cztery kontrolki *Button* zawierające możliwe do wyboru odpowiedzi. Wybranie odpowiedzi powoduje przeładowanie widoku i wyświetlenie na tej samej formatce kolejnego pytania lub gdy było to ostatnie pytanie przeniesienie do trzeciego z kolei widoku aplikacji (Rys. 6).

Rysunek 5: Ekran menu głównego

Rysunek 6: Ekran wygenerowanej diety

Widok ten zawiera grafikę w tle, *TextView* z wygenerowaną dietą oraz *TextBox* wraz z przyciskiem służący do zapisania diety. Na tym etapie użytkownik ma możliwość cofania się do poprzednich pytań i zmiany wybranej odpowiedzi. Aplikacja podczas zapisu sprawdza czy wpisana nazwa figuruje już w bazie, jeśli tak użytkownik jest informowanych o tym fakcie poprzez stosowny komunikat i ma możliwość wprowadzanie nowej, unikalnej nazwy. Istnieje dodatkowe zabezpieczenie, które zapobiega omyłkowemu cofnięciu użytkownika do menu głównego w przypadku, gdy została wygenerowana dieta. Jednokrotne naciśnięcie przycisku wstecz powoduje wyświetlenie alertu o niezapisanej diecie. Użytkownik ma możliwość ponownego naciśnięcia przycisku wstecz i przejścia do ekranu głównego lub zapisu diety. Ostatnim widokiem aplikacji jest widok przeglądania zapisanych diet (Rys. 7). Wejście do tego ekranu odbywa się za pośrednictwem menu głównego po wybraniu opcji Przeglądaj i pod warunkiem, że dostępne są zapisane pozycje. Ekran zawiera Spinner z nazwami zapisanych diet, *ImageButton* przedstawiający kosz na śmieci oraz *TextView* ze składem diety aktualnie wybranego konia. Tło stanowi grafika konia na tle stajni.

Testy Pierwszym rodzajem testów, którym poddawana była aplikacja były testy manualne. Każda wdrażana funkcjonalność została ręcznie sprawdzona pod kątem oczekiwanych lub spodziewanych działań. Wszelkie odstępstwa były analizowane, a po znalezieniu źródła problemu, naprawiane. Po każdym usunięciu błędu testy były powtarzane. Kiedy proces tworzenia aplikacji został zakończony, aplikacja przeszła testy akceptacyjne. Grupa chętnych, potencjalnych użytkowników sprawdzała własnoręcznie możliwości i ograniczenia aplikacji. Zgłoszone uwagi zostały przełożone na usprawnienia aplikacji, tak aby system odpowiadał jak najlepiej na potrzeby osób, które będą go użytkowały.

4. Wnioski

Aplikacja wykorzystuje system ekspertowy do wygenerowania jak najlepiej dopasowanej diety dla określonego konia. Model systemu ekspertowego został zaprojektowany przy wsparciu osób posiadających doświadczenie w dziedzinie hodowli koni, zostały sformułowane pytania oraz zestawy odpowiedzi do każdego z nich, tak aby system mógł pozyskać dane niezbędne do wygenerowania diety.

System został zaimplementowany w języku C#. Bazy wiedzy oraz zawarte w nich reguły zostały przełożone na język obiektowy, został napisany system wnioskujący, którego zadaniem jest wyprowadzenie danych końcowych na ekran użytkownika.

Wybór konkretnej odpowiedzi determinuje ścieżkę do dalszego pytania, w ten sposób ilość zadawanych pytań została ograniczona do minimum. Sugerowane składy dawek pokarmowych z założenia są dawkami dziennymi, zatem można je podzielić na kilka posiłków.

Model jest otwarty na rozszerzenia o kolejne reguły i składy dawek. Zaimplementowanie w sys-

Rysunek 7: Ekran katalogu zapisanych diet

temie zapamiętywania wprowadzonych przez użytkownika odpowiedzi pozwoliłoby stworzyć intuicyjny moduł zmian. Zmian w diecie można by było dokonywać zarówno poprzez ten moduł, edytując poszczególne parametry, np. ilość treningów w tygodniu lub też manualnie zmieniając dawkę z 3 kg na 2 kg. Możliwość zdalnej edycji diety swojego konia zachęcałaby ludzi zapracowanych, niemogących stawić się osobiście.

Literatura

[1] Ł. Bujak, *Drzewa decyzyjne*. Toruń: Uniwersytet Mikołaja Kopernika, 2008.

[2] L. Kiełtyka, "Wykorzystanie systemów eksperckich w zarządzaniu wiedzą," *Zeszyty Naukowe. Organizacja i Zarządzanie / Politechnika Łódzka*, vol. z. 53, 2013.

[3] L. Madeyski, "Xml w bazach danych," *Bazy Danych—Prace Naukowe Wydziałowego Zakładu Informatyki Politechniki Wrocławskiej,(4)*, pp. 81–89, 2003.

[4] K. Michalik, "Pc-shell/sphinx jako narzędzie tworzenia systemów ekspertowych," *Prace Naukowe/Uniwersytet Ekonomiczny w Katowicach*, pp. 35–42, 2010.

[5] D. Modlińska, *Konie i źrebięta : wszystko co o koniu powinien wiedzieć początkujący jeździec i hodowca*. Warszawa: Wydawnictwo Tenten, 1993.

[6] J. Mulawka, *Systemy ekspertowe*. Warszawa: WNT, 1996.

[7] A. Niederliński, *Regułowo-modelowe systemy ekspertowe rmse*. Gliwice: Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, 2006.

[8] I. Rojek, J. Dorożyński, and D. A. Ośka, "Zarządzanie wiedzą przy użyciu systemu ekspertowego," *Studies & Proceedings of Polish Association for Knowledge Management*, vol. 87, pp. 40–49, 2018.

[9] W. Romowicz., *XML. Ćwiczenia praktyczne. Książka*. Gliwice: Helion, 2002. [Online]. Available: https://helion.pl/ksiazki/xml-cwiczenia-praktyczne-wojciech-romowicz,cwxml.htm#format/d

[10] E. Sasimowski and M. Budzyński, *Żywienie koni*. Warszawa: Państwowe Wydawnictwo Rolnicze i Lesne, 1981.

[11] J. Schmuller and K. Masłowski, *UML : Ujednolicony Język Modelowania - wyrażanie związków między klasami w projektowaniu obiektowym*. Gliwice: Helion, 2003.

[12] G. Taskos, *Xamarin. Tworzenie aplikacji cross-platform. Receptury*. Gliwice: Helion, 2017.

[13] K. Waćkowski, J. Krawiec, J. Bereda, E. Chmielewska, and M. Malińska, *Informatyka: terminologia znormalizowana i wykaz norm*. Warszawa: Polski Komitet Normalizacyjny, 2006.

[14] T. Wieczorek, S. Golak, R. Przyłucki, and M. Pilarczyk, "Innowacyjny system ekspertowy do oceny kopalń węgla kamiennego," *Logistyka*, no. 5, CD 1, pp. 645–650, 2015.

[15] R. Wieleba, "Inżynieria wiedzy w systemach ekspertowych," *Zeszyty Naukowe Warszawskiej Wyższej Szkoły Informatyki*, 2011.