

POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY
INSTYTUT ELEKTROTECHNIKI TEORETYCZNEJ
I SYSTEMÓW INFORMACYJNO-POMIAROWYCH
PRACA DYPLOMOWA MAGISTERSKA
Na kierunku **INFORMATYKA**



Krzysztof Kozłowski
Nr indeksu 188140

Rok akademicki: 2006/2007
Warszawa, 26 listopada 2007

System obsługi zawodów sportowych agility

Zakres pracy:

1. Analiza potrzeb użytkownika
2. Zaprojektowanie systemu
3. Wykonanie dwóch aplikacji wchodzących w skład systemu
4. Dokumentacja

*(Podpis i pieczęć
Kierownika Zakładu
Dydaktycznego)*

Opiekun naukowy: dr inż. Jacek Korytkowski
Recenzent: prof. dr hab. Kazimierz Mikołajuk

Termin wykonania: 26 listopada 2007

Spis treści

1	Wstęp	1
1.1	Cel pracy	1
1.2	Aplikacja AgileWeb – użyte narzędzia	2
1.2.1	Język programowania PHP	2
1.2.2	Framework PHP CodeIgniter	2
1.2.3	XAJAX	3
1.3	Aplikacja AgileDesk – użyte narzędzia	3
1.3.1	Język programowania Python	3
1.3.2	Biblioteka interfejsu graficznego GTK+	4
1.4	Słownik użytych w pracy pojęć związanych z zawodami Agility	4
1.5	Priorytety wymagań funkcjonalnych	5
2	Projekt aplikacji AgileWeb	6
2.1	Opis systemu	6
2.1.1	Wizja systemu	6
2.1.2	Funkcjonalne wymagania odnośnie systemu	6
2.1.3	Opis procesów biznesowych	7
2.2	Wymagania funkcjonalne – przypadki użycia i scenariusze	10
2.2.1	Aktorzy	10
2.2.2	Logowanie	10
2.2.3	Zarządzanie kontem	15
2.2.4	Wyświetlenie zawodów	18
2.2.5	Zgłoszenie do zawodów	20
2.2.6	Zarządzanie zawodami	22
2.2.7	Modyfikacja zawodów	24
2.2.8	Zarządzanie użytkownikami	28
2.2.9	Wyświetlenie listy startowej	31
2.2.10	Zarządzanie zgłoszeniami	33
2.3	Wymagania funkcjonalne – reguły biznesowe	35
2.4	Wymagania pozafunkcjonalne	38
2.5	Ograniczenia środowiskowe	39

2.6	Diagram komponentów	39
2.7	Diagram związków encji	40
3	Aplikacja AgileWeb	42
3.1	Instalacja	42
3.1.1	Wymagania	42
3.1.2	Instalacja plików	43
3.1.3	Baza danych	44
3.2	Instrukcja użytkownika	44
3.2.1	Rozpoczęcie pracy – konto	44
3.2.2	Udział w zawodach	46
3.2.3	Zarządzanie zawodami (administrator)	47
3.2.4	Lista startowa (administrator)	49
3.2.5	Wyniki zawodów (administrator)	50
4	Projekt aplikacji AgileDesk	51
4.1	Opis systemu	51
4.1.1	Wizja systemu	51
4.1.2	Funkcjonalne wymagania odnośnie systemu	51
4.1.3	Opis procesów biznesowych	52
4.2	Wymagania funkcjonalne – przypadki użycia i scenariusze	54
4.2.1	Aktorzy	54
4.2.2	Lista startowa	54
4.2.3	Wyniki biegów	56
4.2.4	Obsługa stanu	58
4.3	Wymagania funkcjonalne (reguły biznesowe)	60
4.4	Wymagania pozafunkcjonalne	63
4.5	Ograniczenia środowiskowe	63
4.6	Diagram klas	64
5	Aplikacja AgileDesk	65
5.1	Instalacja	65
5.1.1	Wymagania	65
5.1.2	Instalacja wymaganych komponentów na GNU/Linux	66
5.1.3	Instalacja wymaganych komponentów na Microsoft Windows	66
5.1.4	Konfiguracja AgileDesk	67
5.2	Instrukcja użytkownika	67
5.2.1	Uruchomienie	67
5.2.2	Rozpoczęcie pracy	67
5.2.3	Wyświetlenie i modyfikacja listy startowej	68

5.2.4	Wprowadzanie wyników biegów	69
5.2.5	Wyniki zawodów i eksport	70
6	Specyfikacje plików XML do przechowywania i wymiany danych	71
6.1	Lista startowa	71
6.1.1	Użycie	71
6.1.2	XML Schema	71
6.1.3	Przykład wygenerowanego pliku	73
6.2	Wyniki zawodów	75
6.2.1	Użycie	75
6.2.2	XML Schema	75
6.2.3	Przykład wygenerowanego pliku	76
6.3	Stan systemu AgileDesk	77
6.3.1	Użycie	77
6.3.2	XML Schema	77
6.3.3	Przykład wygenerowanego pliku	80
7	Podsumowanie	82
7.1	Ocena realizacji wymagań	82
7.2	Rozwój systemu	83
	Spis rysunków	84
	Spis tabel	86
	Bibliografia	87

Rozdział 1

Wstęp

1.1 Cel pracy

Celem pracy było zaprojektowanie i wykonanie systemu do obsługi zawodów sportowych psów agility. Omówione niżej założenia i wymogi stawiane systemowi wymusiły podział na dwie osobne aplikacje, nazwane *AgileWeb* i *AgileDesk*.

Podział ten wynikał z formy organizowania zawodów. Pierwszym ich etapem jest rejestracja uczestników, czyli przyjmowanie zgłoszeń do konkretnych zawodów. Rejestracja odbywa się poprzez internet. Po zakończeniu przyjmowania nowych zgłoszeń tworzona jest lista startowa określająca kolejność startów zawodników na samych zawodach. Ten etap został przydzielony aplikacji internetowej *AgileWeb*, dostępnej z poziomu przeglądarki WWW.

Sam przebieg zawodów, względem wymogów stawianych systemowi, podzielić można na:

- dokonywanie ostatecznych zmian w liście startowej oraz wyświetlanie jej w nowej, zmienionej postaci,
- wprowadzanie wyników przebiegów zawodników,
- tworzenie klasyfikacji końcowej zawodów.

Powyższe zadania realizować miała druga aplikacja – desktopowa *AgileDesk* pracująca pod systemami operacyjnymi Microsoft Windows oraz GNU/Linux z graficznym interfejsem użytkownika. Wyniki zakończonych zawodów powinny zostać udostępnione publicznie poprzez odpowiednią stronę WWW w internecie – ta ostatnia wymagana funkcjonalność przydzielona została aplikacji *AgileWeb*. Podstawowym problemem przy takim podziale zadań w

systemie jest wymiana danych pomiędzy dwoma aplikacjami. Z racji organizowania zawodów sportowych w plenerze należało uwzględnić istotne ograniczenie: brak dostępu do internetu podczas ich trwania.

Podsumowując zadania, zaprojektowany i wykonany system ma zatem:

- wspomóc i ujednoczyć rejestrację uczestników na zawody,
- usprawnić zarządzanie zgłoszeniami,
- pomóc organizatorom przy tworzeniu listy startowej,
- obsłużyć przyjmowanie wyników podczas zawodów oraz te wyniki usystematyzować,
- automatycznie tworzyć klasyfikację zawodów,
- umożliwić publikację wyników zawodów w internecie.

1.2 Aplikacja AgileWeb – użyte narzędzia

1.2.1 Język programowania PHP

Aplikacja internetowa AgileWeb została napisana w języku PHP [1] (wersji 5.x). PHP to obecnie jeden z najpopularniejszych języków używanych przy budowie serwisów WWW. Rozprowadzany jest na licencji PHP License [2], uznawanej za fundację Free Software Foundation [3] za Wolne Oprogramowanie [4]. PHP tworzony jest na systemy operacyjne Unix, GNU/Linux oraz Microsoft Windows, co w połączeniu z jego powszechną obsługą przez usługi hostingowe, daje możliwość używania aplikacji w bardzo różnych środowiskach.

1.2.2 Framework PHP CodeIgniter

CodeIgniter [5] to framework PHP wspierający programistę w tworzeniu zaawansowanych aplikacji internetowych. Dystrybuowany jest na własnej licencji, podobnej do licencji Apache lub BSD, zaliczanej do Wolnego Oprogramowania [6]. Cechy CodeIgnitera to m.in.:

- małe rozmiary i związane z tym niewygórowane wymagania wobec środowiska, na którym ma pracować,
- szybka i prosta instalacja,
- bazowanie na wzorcu projektowym MVC (Model-View-Controller),

- wsparcie dla różnych systemów zarządzania bazą danych (DBMS), tj.: MySQL oraz MySQLi, MS SQL, PostgreSQL, Oracle, a także sterowników SQLite i ODBC, obsługa transakcji,
- bogata biblioteka klas oraz funkcji opakowujących typowe funkcje PHP w prostsze w używaniu komponenty.

Wzorzec projektowy MVC (Model-View-Controller) opiera się na trzech warstwach w projektowanej aplikacji:

Model to warstwa odpowiedzialna za przechowywanie, modyfikowanie i wprowadzanie danych oraz za operacje na nich (*logika biznesowa*).

View , czyli widok, to warstwa prezentacji. W przypadku aplikacji internetowej są to generowane dokumenty XHTML.

Controller to warstwa spajająca wcześniejsze dwie poprzez przyjmowanie żądań użytkownika, operowanie na danych w *modelu* i wyświetlanie wyników operacji w wybranym *widoku*.

Zastosowanie wzorca MVC pozwala odseparować poszczególne komponenty odpowiedzialne za obsługę danych, warstwę prezentacji oraz interakcję z użytkownikiem. Dzięki temu usprawnia się proces pisania aplikacji, a tworzonym kodem łatwiej jest zarządzać i później go rozwijać.

1.2.3 XAJAX

W aplikacji AgileWeb wykorzystywana jest technologia AJAX (Asynchronous JavaScript and XML [8]) w celu budowy kilku elementów interakcji z użytkownikiem. Dzięki AJAX-owi aplikacja internetowa osiąga responsywność na żądania użytkownika zbliżoną w odczuciach do aplikacji desktopowej. Podnosi tym samym użyteczność z punktu widzenia typowego użytkownika systemu. Użyta została w tym celu biblioteka XAJAX [9], którą łatwo dołączyć do frameworka CodeIgniter (rozprowadzana na licencji BSD [10]).

1.3 Aplikacja AgileDesk – użyte narzędzia

1.3.1 Język programowania Python

Aplikacja AgileDesk jest typowym przedstawicielem oprogramowania desktopowego. Pracuje pod kontrolą popularnego systemu operacyjnego, a interakcja z użytkownikiem odbywa się poprzez intuicyjny graficzny interfejs

użytkownika (GUI). Jednym z założeń przy jej budowie była możliwość pracy pod różnymi systemami operacyjnymi, ze szczególnym naciskiem na Microsoft Windows oraz GNU/Linux.

Wymaganie to zostało spełnione poprzez użycie języka Python [11] do tworzenia aplikacji (w wersji 2.x). Python to interpretowany język programowania rozwijany na własnej licencji Python Software Foundation License [12] (uznawanej za fundację Free Software za licencję Wolnego Oprogramowania [13]). Jedną z wyróżniających cech Pythona jest minimalistyczna składnia niewymuszająca jednego stylu programowania (tj. możliwe jest programowanie obiektowe, strukturalne oraz funkcyjne). Posiada bardzo bogatą standardową bibliotekę, automatyczne zarządzanie pamięcią (poprzez garbage collector) oraz w pełni dynamiczny system typów.

Interpreter Pythona oraz standardowe biblioteki dostępne są dla większości popularnych systemów operacyjnych, w tym m.in.: Microsoft Windows, GNU/Linux, BSD i Mac OS X. Dzięki temu tworzona przez programistę aplikacja jest w pełni przenośna pomiędzy danymi platformami (o ile wszystkie użyte biblioteki mają swoją implementację na danej platformie).

1.3.2 Biblioteka interfejsu graficznego GTK+

GTK+ [15] (pełna nazwa: The GIMP Toolkit) to biblioteka do tworzenia graficznych interfejsów użytkownika, rozprowadzana na licencji Wolnego Oprogramowania LGPL 2 (Library General Public License) [16]. Bibliotekę można używać z poziomu wielu języków programowania, jak np. C++, C#, Python, PHP, Java, Perl i innych. Dostępna jest dla m.in. systemów Microsoft Windows, BSD, GNU/Linux, Mac OS X.

W celu użycia GTK+ z poziomu aplikacji napisanej w Pythonie należy posłużyć się PyGTK [18] opakowującym funkcje biblioteczne GTK+. PyGTK jest przenośne pomiędzy różnymi platformami na tyle, na ile przenośne są same GTK+ oraz Python.

1.4 Słownik użytych w pracy pojęć związanych z zawodami Agility

Zawody – pojedyncza impreza sportowa.

Zawodnik – para *przewodnik–pies* biorąca udział w zawodach.

Konkurencja – podstawowa jednostka w zawodach. Konkurencja może wyznaczać określony rodzaj i ułożenie toru przeszkód oraz wymagania odnośnie kategorii wzrostowej. Dwie konkurencje mogą być łączone.

Konkurencja łączona – konkurencja składająca się z dwóch innych, tylko

do celów klasyfikacji końcowej zawodów. W konkurencji łączonej uwzględnione są tylko biegi zakończone pomyślnie (bez dyskwalifikacji) z obydwu konkurencji–składowych. Wyniki danego zawodnika, tj. czas i punkty, w takiej konkurencji łączonej są sumą wyników z poszczególnych składowych.

Bieg – podstawowa jednostka w konkurencji – bieg jednego zawodnika.

Numer startowy – jednoznaczny numer przypisany do danego zawodnika.

Kategoria wzrostowa – podział zawodników w konkurencji w zależności od wzrostu psa. Kategoria wzrostowa określa również minimalną wysokość przeszkód. Różne konkurencje mogą mieć różne kategorie wzrostowe.

Organizator – podmiot organizujący i zajmujący się obsługą techniczną zawodów.

Lista startowa – dokument opisujący kolejność startu zawodników podczas jednych zawodów. Określa również kolejność konkurencji i kategorii wzrostowych.

1.5 Priorytety wymagań funkcjonalnych

W projekcie systemu wszystkim wymaganiom funkcjonalnym w projekcie systemu został przypisany jeden z priorytetów:

wysoki (W): wymaganie kluczowe, podstawowe dla funkcjonalności aplikacji;

średni (Ś): wymaganie istotne dla funkcjonalności aplikacji; jego brak lub niepełne spełnienie jest w ograniczonym zakresie dopuszczalne;

niski (N): wymaganie opcjonalne, podnoszące jakość lub rozszerzające funkcjonalność; nieobowiązkowe; ewentualnie planowane do zrealizowania w przyszłości;

Rozdział 2

Projekt aplikacji AgileWeb

2.1 Opis systemu

2.1.1 Wizja systemu

System ma na celu rozwiązanie problemów związanych z zapisami i obsługą uczestników podczas zawodów. Dzięki wykorzystaniu systemu:

- zostanie usprawniony i ułatwiony proces rejestracji zawodników,
- organizator uzyska scentralizowany mechanizm zarządzania zawodnikami,
- uproszczony zostanie przygotowywanie list startowych i przetwarzanie wyników zawodów.

2.1.2 Funkcjonalne wymagania odnośnie systemu

Konto użytkownika – umożliwienie użytkownikowi utworzenie konta w systemie (tym samym zostanie użytkownikiem systemu), a także wprowadzenie swoich danych osobowych i ich modyfikację.

Obsługa zgłoszeń – przyjmowanie, modyfikację i wycofanie zgłoszeń przez użytkowników.

Zarządzanie użytkownikami – usuwanie, dodawanie i modyfikacja użytkowników systemu przez upoważnioną osobę.

Zarządzanie zgłoszeniami – usuwanie i modyfikacja zgłoszeń użytkowników przez upoważnioną osobę.

Zarządzanie zawodami – wyświetlanie, tworzenie, edycja i kasowanie zawodów.

Obsługa list startowych – wyświetlanie (również w postaci przeznaczonej

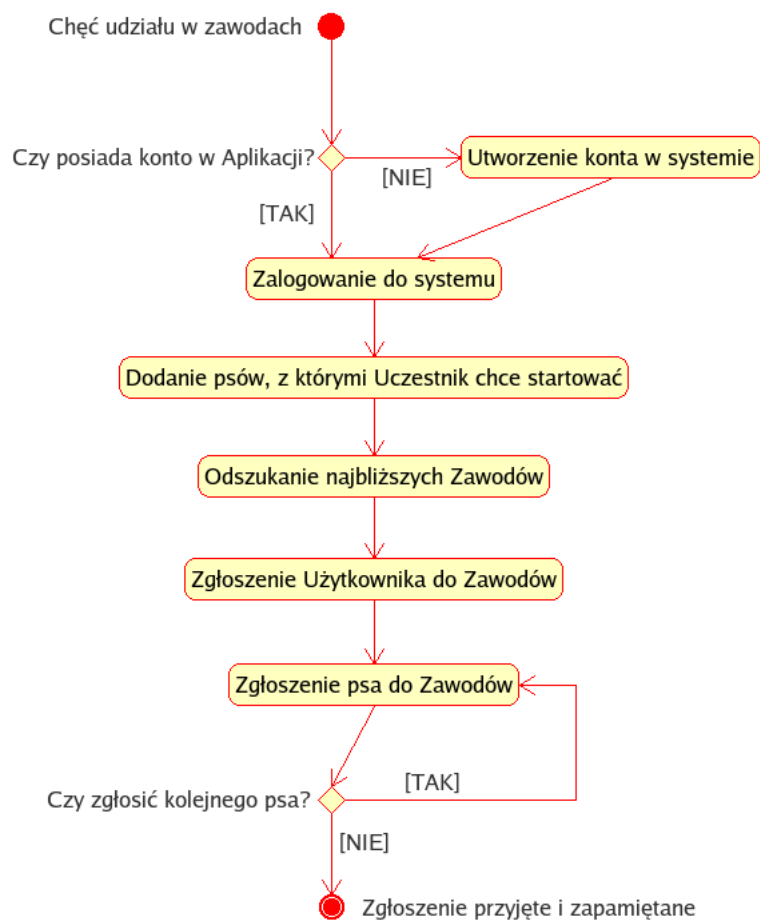
do druku) i eksport do pliku XML list startowych.

Obsługa wyników – import wyników zawodów z pliku XML do systemu.

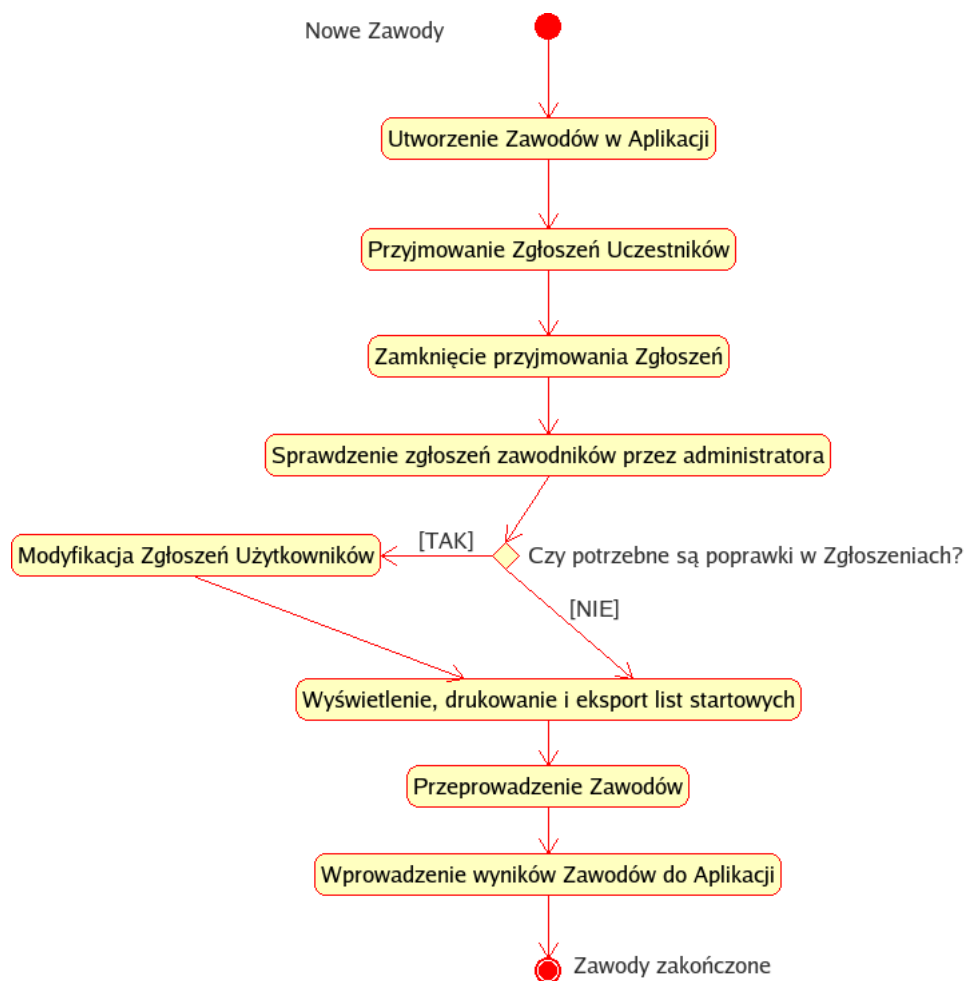
2.1.3 Opis procesów biznesowych

Diagram 2.1 przedstawia schemat procesu zgłaszania się uczestnika do zawodów. Przede wszystkim cały proces bazuje na obowiązku zalogowania się do systemu na swoje konto (w przypadku braku konta – wpierw musi być utworzone). Z kontem związane są psy użytkownika, z którymi może startować w konkretnych zawodach.

Diagram 2.2 opisuje z kolei proces pojedynczych zawodów z punktu widzenia czynności związanych z aplikacją AgileWeb. Zwrócić należy uwagę na konieczność sprawdzenia zgłoszeń przez administratora. Same zawody zostały tu ujęte w pojedynczą akcję *Przeprowadzenie Zawodów* jako, że aplikacja w żaden sposób nie jest przywiązana do ich konkretnej formy.



Rysunek 2.1: Zgłoszenie w zawodach – schemat

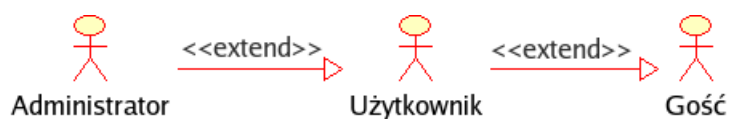


Rysunek 2.2: Przebieg zawodów – schemat

2.2 Wymagania funkcjonalne – przypadki użycia i scenariusze

2.2.1 Aktorzy

- **Gość** – Dowolna osoba niezalogowana w systemie. Dostęp tylko do części aplikacji związanych z przypadkami użycia: [rejestracja], [logowanie], [aktywacja konta], [przypomnienie hasła], [wyświetlenie listy zawodów], [wyświetlenie zawodów], [wyświetlenie wyników zawodów], [wyświetlenie list startowych].
- **Użytkownik** – Osoba, która założyła konto w systemie, tym samym może zarządzać tym kontem oraz zgłaszać uczestnictwo w zawodach.
- **Administrator** – Upoważniona przez Organizatora osoba, posiadająca konto w systemie o specjalnych przywilejach z dostępem do wszystkich przypadków użycia i danych w systemie.



Rysunek 2.3: AgileWeb – Aktorzy w systemie

2.2.2 Logowanie

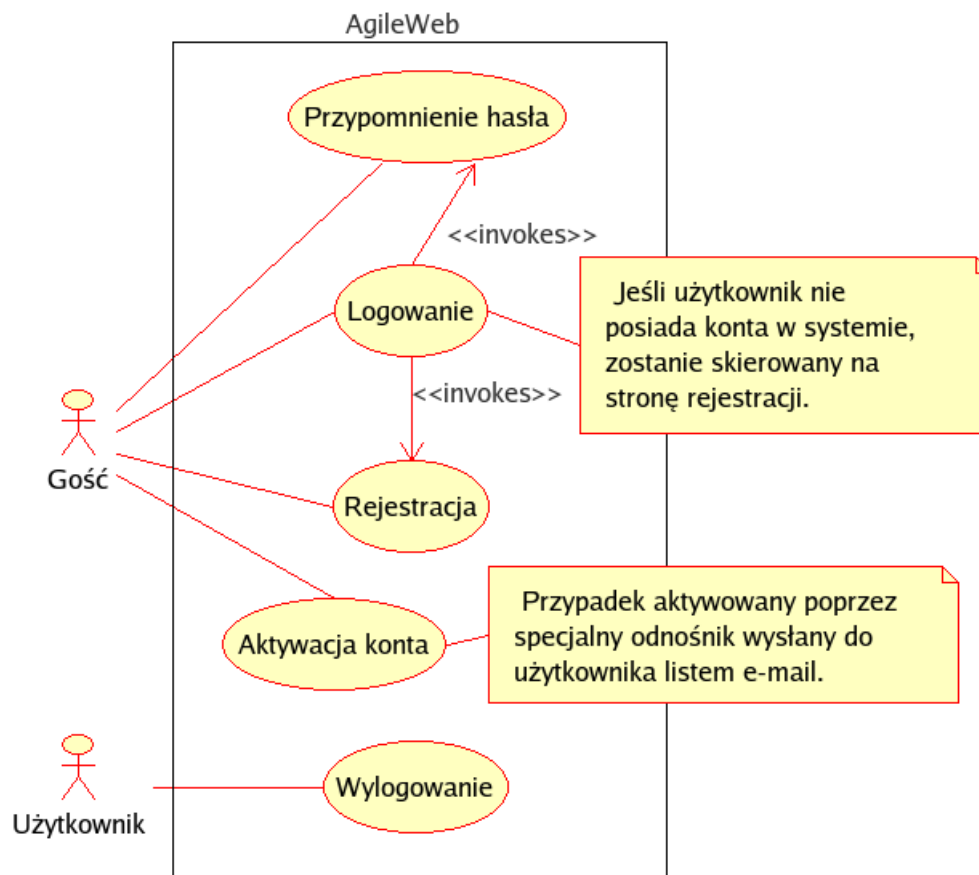
Zalogowanie się

Priorytet: wysoki

Aktorzy: gość

Opis:

1. Wyświetlony zostaje [ekran logowania].
2. [Gość] wypełnia pola formularza ([nazwa użytkownika] oraz [hasło]) i naciska przycisk *zaloguj*.
3. Jeśli dane uwierzytelniające były poprawne [gość] zostaje zalogowany. Wyświetlony zostaje komunikat o poprawnym logowaniu.
4. Odnośnik *Zaloguj się* (w menu głównym) zmieniony zostaje w *Wyloguj się*.



Rysunek 2.4: AgileWeb – przypadek użycia – Logowanie

Alternatywne zakończenie scenariusza:

1. Jeśli dane uwierzytelniające nie były poprawne, to wyświetlony zostaje komunikat o błędzie.
2. Jeśli ekran został wywołany przez [użytkownika] zalogowanego, to wyświetlony zostaje komunikat o błędzie.

Dodatkowe informacje: [System] nie informuje, które dane były niepoprawne – czy [nazwa użytkownika] czy [hasło]. Każdy z przypadków nieprawidłowego logowania oznajmiany jest tym samym komunikatem błędu.

Rejestracja

Priorytet: wysoki

Aktorzy: gość

Opis:

1. [Gość] uruchamia [ekran rejestracji].
2. [Gość] wypełnia pola formularza ([nazwa użytkownika], dwa pola [hasło], dwa pola [adres e-mail]) i naciska przycisk *Zalóż konto*.
3. Jeśli formularz został poprawnie wypełniony oraz [system] pozwala na założenie konta, to:
 - (a) tworzone jest nowe konto,
 - (b) [gość] automatycznie jest logowany,
 - (c) wysyłany jest list e-mail aktywujący konto.

Alternatywne zakończenie scenariusza:

1. Jeśli użytkownik wprowadził różne [hasła] lub [adresy e-mail], nie wypełnił wszystkich obowiązkowych pól lub wprowadził nieprawidłowe znaki, to wyświetlony zostaje komunikat o błędzie.
2. Jeśli w systemie istnieje [użytkownik] o takiej samej nazwie jak wprowadzona w formularzu [nazwa użytkownika], to wyświetlony zostaje komunikat o błędzie.

Dodatkowe informacje: Wymaganymi polami w formularzu rejestracyjnym są:

imię – zawierać może dowolne znaki drukowalne lub białe;

nazwisko – zawierać może dowolne znaki drukowalne lub białe;

nazwa użytkownika – zawierać może znaki alfanumeryczne (z alfabetu łacińskiego i polskiego), podkreślenie, kropkę, myślnik; unikalne na całą bazę, tj. może istnieć maksymalnie jedno konto o danej [nazwie użytkownika];

hasło – zawierać może dowolne znaki; nie może być krótsze od 7 znaków;

adres e-mail – zawierać może tylko znaki dozwolone w adresach e-mail i składać się z dwóch członów rozdzielonych znakiem @ – członu użytkownika (dozwolone: znaki alfanumeryczne z alfabetu łacińskiego, podkreślenie, kropka, myślnik) oraz domeny (dozwolone: znaki alfanumeryczne z alfabetu łacińskiego, kropka, myślnik); adres e-mail nie musi być unikalny w bazie, tj. wielu użytkowników może mieć ten sam adres.

Opcjonalnymi polami w formularzu rejestracyjnym są:

nazwa klubu – zawierać może znaki alfanumeryczne (z alfabetu ła-
cińskiego i polskiego), podkreślenie, kropkę, myślnik;
telefon – zawierać może tylko cyfry lub znaki białe;
ulica – dowolny napis;
kod pocztowy – zawierać może tylko cyfry i myślnik;
miasto – dowolny napis;
junior – wartość boolowska;
debiutant – wartość boolowska.

Wylogowanie się

Priorytet: wysoki

Aktorzy: użytkownik, administrator

Opis:

1. [Użytkownik] uruchamia akcję wylogowania.
2. [System] usuwa sesję użytkownika.
3. Wyświetlony zostaje komunikat o pomyślnym wylogowaniu się.
4. Odnośnik *Wyloguj się* (w menu głównym) zmieniony zostaje w *Zaloguj się*.

Alternatywne zakończenie scenariusza: Jeśli [system] nie mógł usunąć sesji użytkownika, to wyświetlony zostaje komunikat o błędzie.

Aktywacja konta

Priorytet: średni

Aktorzy: użytkownik

Opis:

1. [Użytkownik] wchodzi na stronę aktywacji konta poprzez odnośnik, który otrzymał wcześniej listem e-mail.
2. Wyświetlony zostaje komunikat o pomyślnej aktywacji konta i [system] automatycznie loguje [użytkownika].

Alternatywne zakończenie scenariusza: [System] wyświetli komunikat o błędzie, jeśli dane aktywacyjne nie są prawidłowe.

Dodatkowe informacje: Odnośnik do aktywacji konta zawierać musi losowo wygenerowany identyfikator (ciąg znaków alfanumerycznych), po którym [system] rozpozna konto podlegające aktywacji. Identyfikator nie może być przewidywalny.

Przypomnienie hasła

Priorytet: średni

Aktorzy: gość

Opis:

1. [Gość] uruchamia [ekran przypomnienia hasła].
2. [Gość] wypełnia pola formularza ([nazwa użytkownika] oraz [adres e-mail]) i naciska przycisk *Wyślij nowe hasło*.
3. System szuka konta w bazie o podanej [nazwie użytkownika] z podanym [adresem e-mail].
4. Jeśli system odnalazł pasujące konto, to:
 - (a) tworzy nowe losowe [hasło] i zapisuje je w bazie jako [tymczasowe hasło] wymagające potwierdzenia (nienadpisując oryginalnego hasła użytkownika);
 - (b) wysyła list e-mail na dany adres z wylosowanym właśnie [tymczasowym hasłem];
 - (c) [użytkownik] wchodzi na stronę logowania i loguje się używając otrzymanego [tymczasowego hasła];
 - (d) wyświetlony zostaje komunikat o pomyślnym logowaniu i ustawieniu nowego hasła [użytkownika] na [hasło tymczasowe].

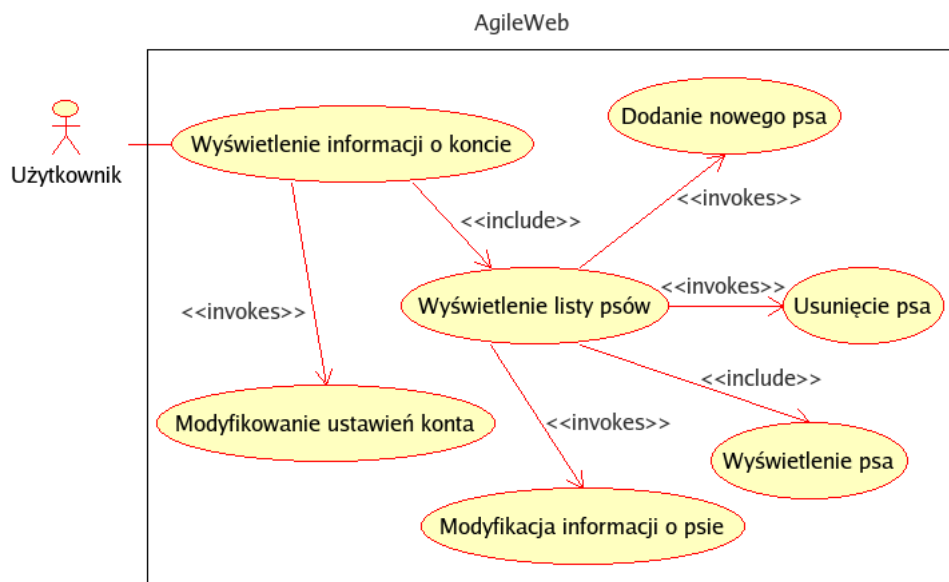
Alternatywne zakończenie scenariusza: [System] wyświetli komunikat o błędzie, jeśli nie odnalazł w bazie pasującego konta do podanego [adresu e-mail] i [nazwy użytkownika].

Dodatkowe informacje: Wylosowane hasło składać się może z:

- małych i wielkich liter alfabetu łacińskiego (bez znaków narodowych);
- cyfr.

i musi mieć długość 8 znaków.

2.2.3 Zarządzanie kontem



Rysunek 2.5: AgileWeb – przypadek użycia – Zarządzanie kontem

Wyświetlenie informacji o koncie

Priorytet: wysoki

Aktorzy: użytkownik

Opis:

1. [Użytkownik] wchodzi do ustawień jego konta – wyświetlony zostaje [ekran informacji o użytkowniku].
2. Ekran zawiera wszystkie dane użytkownika (wymienione w wymaganiu [(Rejestracja)]) poza hasłem.
3. Wyświetlone zostają dodane przez [użytkownika] [psy] wraz z informacjami o nich ([imię psa], [imię rodowodowe psa], [rasa psa]).

Modyfikowanie ustawień konta

Priorytet: wysoki

Aktorzy: użytkownik

Opis:

1. Wyświetlony zostaje [ekran modyfikacji konta użytkownika].
2. [Użytkownik] zmienia dane znajdujące się w formularzu.
3. [Użytkownik] naciska przycisk *Zapisz zmiany*.
4. [System] wprowadza zmiany użytkownika i informuje go o pomyślnym wykonaniu operacji.
5. Zostaje wyświetlony [ekran informacji o użytkowniku].

Alternatywne zakończenie scenariusza: Jeśli [użytkownik] nie wypełnił wszystkich obowiązkowych pól lub wprowadził nieprawidłowe znaki, to wyświetlony zostaje komunikat o błędzie i ponownie [ekran modyfikacji konta użytkownika].

Dodatkowe informacje: Zmianie nie podlega [nazwa użytkownika].

Dodanie nowego psa

Priorytet: wysoki

Aktorzy: użytkownik

Opis:

1. Wyświetlony zostaje [ekran modyfikacji psa] z pustymi polami formularza.
2. [Użytkownik] wprowadza informacje o [psie] w formularzu.
3. [Użytkownik] naciska przycisk *Dodaj psa*.
4. [System] dodaje nowego [psa] i informuje [użytkownika] o pomyślnym wykonaniu operacji.
5. Zostaje wyświetlony [ekran informacji o użytkowniku].

Alternatywne zakończenie scenariusza: Jeśli [użytkownik] nie wypełnił wszystkich obowiązkowych pól lub wprowadził nieprawidłowe znaki, to wyświetlony zostaje komunikat o błędzie oraz ponownie [ekran modyfikacji psa].

Dodatkowe informacje: Obowiązkowymi polami są:

- [imię psa] (napis)
- [rasa psa] (napis)

- [wzrost] (liczba całkowita)
- [data urodzenia] (data)

Nieobowiązkowym polem są: [imię rodowodowe psa] (napis), [płeć] (wybór: pies lub suka), [weteran] (wartość boolowska).

Modyfikacja informacji o psie

Priorytet: wysoki

Aktorzy: użytkownik

Opis:

1. Wyświetlony zostaje [ekran modyfikacji psa] z wypełnionymi polami formularza (danymi [psa]).
2. [Użytkownik] zmienia informacje o [psie] w formularzu.
3. [Użytkownik] naciska przycisk *Zapisz zmiany*.
4. [System] modyfikuje dane psa i informuje [użytkownika] o pomyślnym wykonaniu operacji.
5. Zostaje wyświetlony [ekran informacji o użytkowniku].

Alternatywne zakończenie scenariusza: Jeśli [użytkownik] nie wypełnił wszystkich obowiązkowych pól lub wprowadził nieprawidłowe znaki, to wyświetlony zostaje komunikat o błędzie i ponownie [ekran modyfikacji psa].

Dodatkowe informacje: Obowiązkowe i opcjonalne pola wymienione są w wymaganiu [(Dodanie nowego psa)].

Usunięcie psa

Priorytet: wysoki

Aktorzy: użytkownik

Opis:

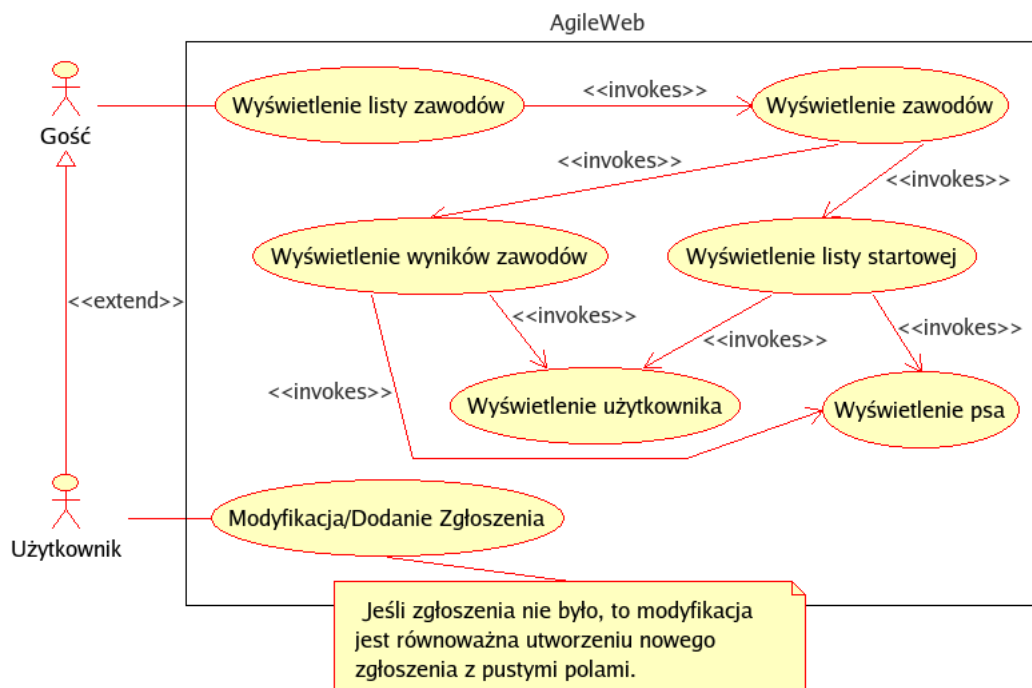
1. [System] usuwa wybranego [psa] i informuje [użytkownika] o pomyślnym wykonaniu operacji.
2. Zostaje wyświetlony [ekran informacji o użytkowniku].

Alternatywne zakończenie scenariusza: Jeśli [system] nie usunął psa, to wyświetlony zostaje komunikat o błędzie.

Dodatkowe informacje: Usunięcie [psa] nie wiąże się z fizycznym skasowaniem rekordu z bazy danych, ale tylko z zaznaczeniem go jako *usuniętego*:

- Dalej dotyczyć go będą przypadki użycia: [(Wyświetlenie wyników zawodów)], [(Wyświetlenie historii startów)] (ale tylko historii użytkownika).
- Nie będzie można go przywrócić do poprzedniego stanu.
- Nie będą dotyczyć go inne przypadki użycia.

2.2.4 Wyświetlenie zawodów



Rysunek 2.6: AgileWeb – przypadek użycia – Wyświetlenie zawodów

Wyświetlenie listy zawodów

Priorytet: wysoki

Aktorzy: gość, użytkownik, administrator

Opis:

1. Wyświetlony zostaje [ekran listy zawodów].
2. Dla aktora [użytkownik] przy każdej pozycji w liście zawodów widnieje odnośnik do przejścia do przypadków użycia [(Dodanie zgłoszenia)] lub [(Modyfikacja zgłoszenia)].
3. Dla aktora [administrator] przy każdej pozycji w liście zawodów widnieje odnośnik do przejścia do przypadku użycia [(Wyświetlenie zgłoszeń w zawodach)].

Wyświetlenie zawodów

Priorytet: wysoki

Aktorzy: gość, użytkownik, administrator

Opis:

1. Użytkownik wybiera [zawody] i wyświetlony zostaje [ekran informacji o zawodach].
2. Dla aktora [użytkownik] widnieje odnośnik do przejścia do przypadków użycia [(Dodanie zgłoszenia)] lub [(Modyfikacja zgłoszenia)].

Wyświetlenie wyników zawodów

Pre: Zawody zostały zakończone (obecna data jest równa dacie zawodów) i zostały wprowadzone wyniki

Priorytet: wysoki

Aktorzy: gość, użytkownik

Opis:

1. Zostaje wyświetlony [ekran wyników zawodów].
2. Dla aktora [użytkownik] przy każdej pozycji w liście wyników widnieje odnośnik do przejścia do przypadków użycia [(Wyświetlenie użytkownika)] lub [(Wyświetlenie psa)] (tylko jeśli docelowy [użytkownik] lub [pies] istnieją).

Wyświetlenie listy startowej

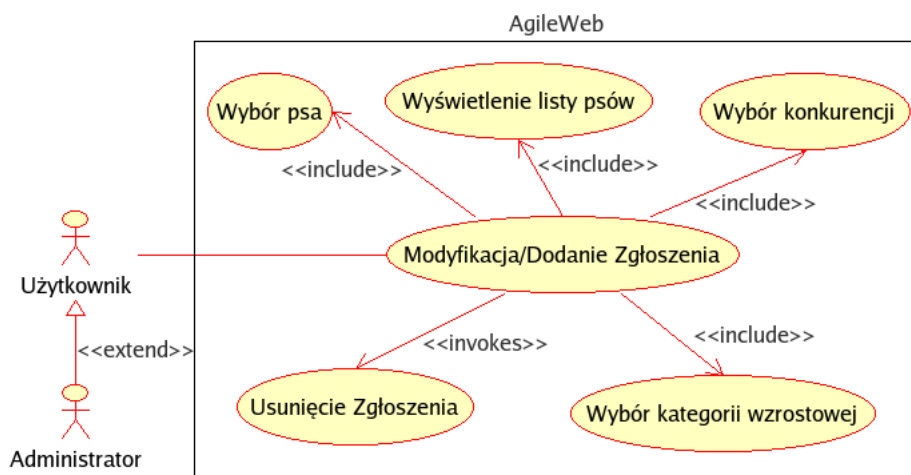
Priorytet: wysoki

Aktorzy: gość, użytkownik, administrator

Opis:

1. Zostaje wyświetlony [ekran listy startowej zawodów].
2. Dla aktora [użytkownik] i [administrator] przy każdej pozycji w liście wyników widnieje odnośnik do przejścia do przypadków użycia [(Wyświetlenie użytkownika)] lub [(Wyświetlenie psa)] (tylko jeśli docelowy [użytkownik] lub [pies] istnieją).

2.2.5 Zgłoszenie do zawodów



Rysunek 2.7: AgileWeb – przypadek użycia – Zgłoszenie do zawodów

Dodanie zgłoszenia

Pre: [(Wyświetlenie zawodów)]

Priorytet: wysoki

Aktorzy: użytkownik, administrator

Opis:

1. Wyświetlony zostaje [ekran modyfikacji zgłoszenia w zawodach] z listą dotychczasowych [zgłoszeń] w tych [zawodach] oraz z formularzem zgłoszeniowym z pustymi polami.
2. [Użytkownik] ustawia [psa], [kategorię wzrostową] i [konkurencję] wybierając jedną pozycję z rozwijanej listy.
3. Po wybraniu pozycji z pola [imię psa] automatycznie powinien uaktualnić się możliwy wybór [konkurencji] bazując na ewentualnych wykluczeniach pomiędzy konkurencjami.
4. Po wybraniu pozycji z pola [konkurencja] automatycznie powinien uaktualnić się możliwy wybór [kategorii wzrostowych] bazując na powiązaniu między [konkurencją] a [kategoriami wzrostowymi].
5. Jeżeli [użytkownik] wybrał inną [kategorię wzrostową] niż tę wynikającą ze wzrostu psa, to system powinien wyświetlić informację o potencjalnym błędzie.
6. [Użytkownik] naciska przycisk *Zgłoś*.
7. Jeżeli [użytkownik] prawidłowo wypełnił formularz, to [system] zapisuje [zgłoszenie] w bazie danych i wyświetlony zostaje komunikat o sukcesie.
8. Wyświetlony zostaje [ekran modyfikacji zgłoszenia w zawodach].

Alternatywne zakończenie scenariusza: Jeśli użytkownik nieprawidłowo wypełnił formularz wprowadzając nieprawidłowe dane lub niepodając obowiązkowych pól, to wyświetlony zostaje komunikat o błędzie oraz ponownie [ekran modyfikacji zgłoszenia w zawodach].

Dodatkowe informacje: Nie jest możliwe dodanie [zgłoszenia] w zakończonych [zawodach].

Modyfikacja zgłoszenia

Priorytet: wysoki

Aktorzy: użytkownik, administrator

Opis:

1. Wyświetlony zostaje [ekran modyfikacji zgłoszenia w zawodach] z listą dotychczasowych [zgłoszeń] w tych [zawodach].

2. [Użytkownik] w wybranym [zgłoszeniu] modyfikuje pola: [pies], [kategoria wzrostowa] lub [konkurencja] wybierając jedną pozycję z rozwijanej listy (reakcja na ich zmianę jest podobna w przypadku [(Dodanie zgłoszenia)]).
3. [Użytkownik] naciska przycisk *Zapisz zmiany*.
4. Jeżeli [użytkownik] prawidłowo wypełnił formularz, to [system] modyfikuje [zgłoszenie] w bazie danych i wyświetlony zostaje komunikat o sukcesie.
5. Wyświetlony zostaje [ekran modyfikacji zgłoszenia w zawodach].

Alternatywne zakończenie scenariusza: Jeśli [użytkownik] nieprawidłowo wypełnił formularz wprowadzając nieprawidłowe dane lub niepodając obowiązkowych pól, to wyświetlony zostaje komunikat o błędzie oraz ponownie [ekran modyfikacji zgłoszenia w zawodach].

Usunięcie zgłoszenia

Priorytet: wysoki

Aktorzy: użytkownik, administrator

Opis:

1. Wyświetlony zostaje [ekran modyfikacji zgłoszenia w zawodach] z listą dotychczasowych [zgłoszeń] w tych [zawodach].
2. [Użytkownik] naciska na przycisk *Usuń zgłoszenie* w wybranym [zgłoszeniu]
3. [System] usuwa dane [zgłoszenie] z bazy danych i wyświetlony zostaje komunikat o sukcesie.
4. Wyświetlony zostaje [ekran modyfikacji zgłoszenia w zawodach].

2.2.6 Zarządzanie zawodami

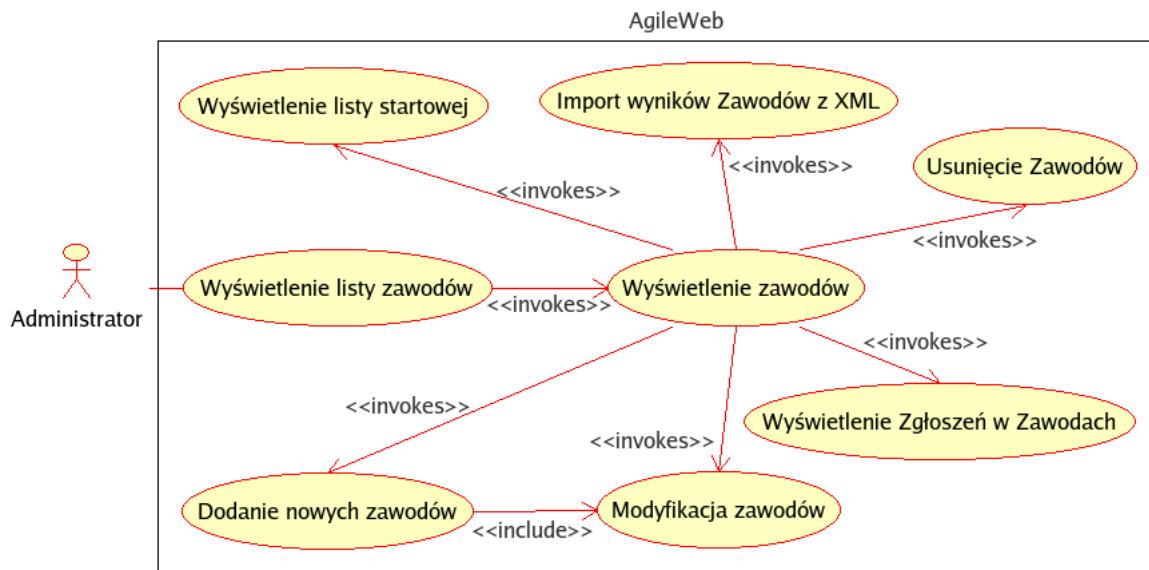
Import wyników zawodów z XML

Pre: Zawody zostały zakończone (obecna data jest równa dacie zawodów)

Priorytet: wysoki

Aktorzy: administrator

Opis:



Rysunek 2.8: AgileWeb – przypadek użycia – Zarządzanie zawodami

1. Wyświetlony zostaje [ekran importowania wyników zawodów].
2. W formularzu [administrator] wskazuje plik XML z lokalnego dysku z [wynikami zawodów].
3. [Administrator] naciska przycisk *Prześlij plik XML*.
4. Następuje przesyłanie pliku poprzez żądanie POST.
5. [System] przetwarza plik XML i wprowadza rekordy do bazy danych.
6. Jeśli było błędów przy parsowaniu pliku i importowaniu rekordów, to wyświetlony zostaje komunikat o sukcesie.
7. Wyświetlony zostaje [ekran wyników zawodów].

Alternatywne zakończenie scenariusza:

1. Jeśli importowany plik XML posiada błędy i operacja importu nie mogła zostać rozpoczęta, to wyświetlony zostaje komunikat o błędzie oraz ponownie [ekran importowania wyników zawodów].
2. Jeśli nastąpił błąd podczas wprowadzania rekordów do bazy (tj. część rekordów została wprowadzona), to wyświetlony zostaje komunikat o błędzie oraz ponownie [ekran wyników zawodów].

Usunięcie zawodów

Priorytet: wysoki

Aktorzy: administrator

Opis:

1. Zostaje wyświetlona informacja ostrzegająca przed usunięciem zawodów. [Użytkownik] może anulować akcję lub potwierdzić ją.
2. Jeśli [użytkownik] potwierdził chęć usunięcia [zawodów], to system usuwa wybrane [zawody] i informuje [użytkownika] o pomyślnym wykonaniu operacji.
3. Zostaje wyświetlony [ekran listy zawodów].

Alternatywne zakończenie scenariusza:

- Jeśli [system] nie usunął zawodów, to wyświetlony zostaje komunikat o błędzie.
- Jeśli użytkownik anulował usuwanie, to wyświetlony zostaje [ekran informacji o zawodach].

Dodatkowe informacje: Usunięcie [zawodów] wiąże się z fizycznym skasowaniem z bazy danych rekordów związanych z tymi zawodami: [zawodów], [zgłoszeń], [list startowych], [wyników zawodów].

2.2.7 Modyfikacja zawodów

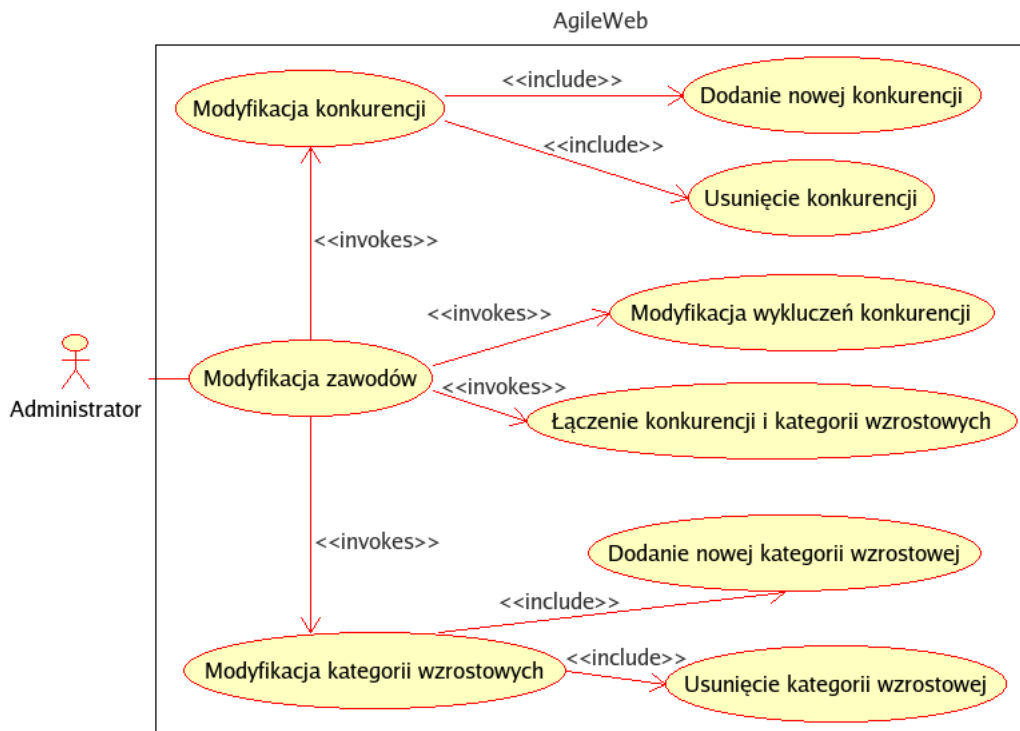
Dodanie nowych zawodów

Priorytet: wysoki

Aktorzy: administrator

Opis:

1. Wyświetlony zostaje [ekran modyfikacji zawodów] z formularzem (z pustymi polami).
2. [Administrator] wypełnia formularz.
3. [Administrator] naciska przycisk *Dodaj zawody*.
4. Jeżeli [administrator] prawidłowo wypełnił formularz, to [system] dodaje nowe [zawody] do bazy danych i wyświetlony zostaje komunikat o sukcesie.



Rysunek 2.9: AgileWeb – przypadek użycia – Modyfikacja zawodów

- Wyświetlony zostaje [ekran informacji o zawodach] z komunikatem o konieczności ustawienia [konkurencji] i [kategorii wzrostowych] w tych [zawodach].

Alternatywne zakończenie scenariusza: Jeśli [administrator] nieprawidłowo wypełnił formularz wprowadzając nieprawidłowe dane lub niepodając obowiązkowych pól, to [wyświetlony zostaje komunikat o błędzie oraz ponownie [ekran modyfikacji zawodów].

Dodatkowe informacje: Obowiązkowymi polami są: [nazwa zawodów] (napis) oraz [data]. Opcjonalnym polem jest [opis] (napis, z możliwym kodem HTML) oraz [konkurencje] przypisane danym zawodom (lista przycisków typu checkbox).

Modyfikacja zawodów

Priorytet: wysoki

Aktorzy: administrator

Opis:

1. Wyświetlony zostaje [ekran modyfikacji zawodów] z formularzem (z wypełnionymi polami).
2. [Administrator] modyfikuje formularz.
3. [Administrator] naciska przycisk *Zapisz zmiany*.
4. Jeżeli [administrator] prawidłowo wypełnił formularz, to [system] modyfikuje [zawody] w bazie danych i wyświetlony zostaje komunikat o sukcesie.
5. Wyświetlony zostaje [ekran informacji o zawodach].

Alternatywne zakończenie scenariusza: Jeśli [administrator] nieprawidłowo wypełnił formularz wprowadzając nieprawidłowe dane lub niepodając obowiązkowych pól, to wyświetlony zostaje komunikat o błędzie oraz ponownie [ekran modyfikacji zawodów].

Dodatkowe informacje: Obowiązkowe i opcjonalne pola [zawodów] wymienione są w wymaganiu [(Dodanie nowych zawodów)].

Łączenie konkurencji i kategorii wzrostowych w zawodach

Priorytet: wysoki

Aktorzy: administrator

Opis:

1. Wyświetlony zostaje [ekran łączenia kategorii wzrostowych i konkurencji w zawodach].
2. Ekran zawiera listę wcześniej ustalonych połączeń ([konkurencja]-[kategoria wzrostowa]) z odnośnikami prowadzącymi do usunięcia każdego z połączeń oraz jedną pustą pozycję na dodanie nowego połączenia.
3. Możliwe są dwa kierunki rozwoju scenariusza:
 - (a) [Administrator] może usunąć jedno połączenie poprzez wejście w wybrany odnośnik *Usuń*.
 - (b) [Administrator] może zmienić pola ustalonych już połączeń i/lub wypełnić pola tworzenia nowego połączenia, na koniec naciska przycisk *Dodaj* lub *Wprowadź zmiany*.

4. [System] usuwa, dodaje lub modyfikuje połączenia w zależności od żądania [Administradora] i wyświetlony zostaje komunikat o sukcesie.
5. Wyświetlony zostaje [ekran modyfikacji kategorii wzrostowych i konkurencji w zawodach].

Alternatywne zakończenie scenariusza: Jeśli [system] nie będzie mógł dokonać modyfikacji połączeń, to wyświetlony zostaje komunikat o błędzie oraz ponownie [ekran modyfikacji kategorii wzrostowych i konkurencji w zawodach].

Modyfikacja wykluczeń konkurencji

Priorytet: średni

Aktorzy: administrator

Opis:

1. Wyświetlony zostaje [ekran wykluczeń konkurencji].
2. Ekran zawiera dokonane wcześniej wykluczenia pomiędzy dwiema [konkurencjami] oraz jedną pustą pozycję na dodanie nowego wykluczenia.
3. Możliwe są dwa kierunki rozwoju scenariusza:
 - (a) [Administrator] może usunąć jedno wykluczenie poprzez wejście w wybrany odnośnik *Usuń*.
 - (b) [Administrator] może zmienić pola ustalonych już wykluczeń i/lub wypełnić pola tworzenia nowego wykluczenia, na koniec naciska przycisk *Dodaj* lub *Wprowadź zmiany*.
4. [System] usuwa, dodaje lub modyfikuje wykluczenia w zależności od żądania [Administradora] i wyświetlony zostaje komunikat o sukcesie.
5. Wyświetlony zostaje [ekran wykluczeń konkurencji].

Alternatywne zakończenie scenariusza: Jeśli nastąpi błąd modyfikacji wykluczeń, to wyświetlony zostaje komunikat o błędzie oraz ponownie [ekran wykluczeń konkurencji].

Modyfikacja konkurencji, modyfikacja kategorii wzrostowej

Priorytet: wysoki

Aktorzy: administrator

Opis:

1. Wyświetlony zostaje [ekran modyfikacji konkurencji] poprzez kliknięcie na odpowiedni odnośnik na [ekranie listy zawodów].
2. Ekran zawiera listę istniejących wszystkich [konkurencji], w której każda pozycja jest edytowalna (pole tekstowe) oraz posiada ikonę usunięcia danej [konkurencji].
3. Możliwe są dwa kierunki rozwoju scenariusza:
 - (a) [Administrator] może usunąć jedną konkurencję poprzez wejście w wybrany odnośnik *Usuń* (zostanie wyświetlona prośba o potwierdzenie usunięcia).
 - (b) [Administrator] może zmienić pola istniejących już konkurencji i/lub wypełnić pola tworzenia nowej konkurencji, na koniec naciska przycisk *Wprowadź zmiany*.
4. [System] usuwa, dodaje lub modyfikuje konkurencje w zależności od żądania [Administradora] i wyświetlony zostaje komunikat o sukcesie.
5. Wyświetlony zostaje [ekran modyfikacji konkurencji].

Alternatywne zakończenie scenariusza: Jeśli [administrator] nieprawidłowo wypełnił formularz wprowadzając nieprawidłowe dane lub niepodając obowiązkowych pól, to wyświetlony zostaje komunikat o błędzie oraz ponownie [ekran modyfikacji konkurencji].

Dodatkowe informacje: Scenariusz dla modyfikacji [kategorii wzrostowej] jest analogiczny.

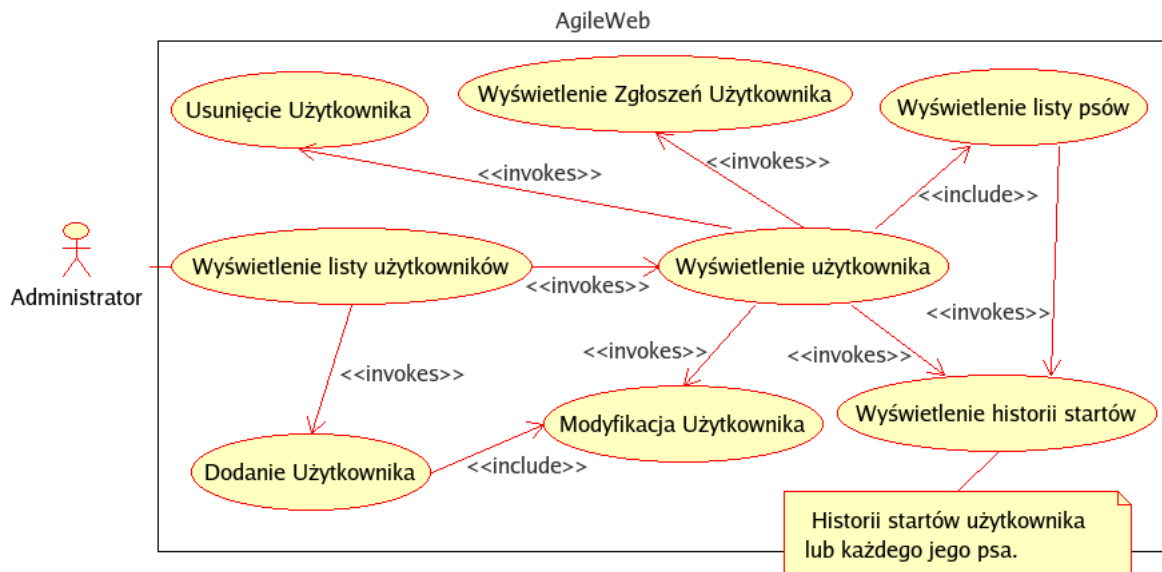
2.2.8 Zarządzanie użytkownikami

Wyświetlenie listy użytkowników

Priorytet: wysoki

Aktorzy: administrator

Opis:



Rysunek 2.10: AgileWeb – przypadek użycia – Zarządzanie użytkownikami

1. Wyświetlony zostaje [ekran listy użytkowników].
2. Przy każdym elemencie listy jest odnośnik do [ekranu informacji o użytkowniku].

Dodatkowe informacje: Lista [użytkowników] posortowana jest po nazwisku (rosnąco). [Administrator] może odwrócić sortowanie.

Wyświetlenie listy psów, Wyświetlenie listy psów użytkownika

Priorytet: wysoki

Aktorzy: użytkownik, administrator

Opis:

1. Wyświetlony zostaje [ekran listy psów].
2. Lista [psów] zawiera [psy] należące do danego [użytkownika],

Wyświetlenie użytkownika

Priorytet: wysoki

Aktorzy: administrator

Opis:

1. Wyświetlony zostaje [ekran informacji o użytkowniku].
2. Ekran zawiera również [listę psów] danego [użytkownika].

Alternatywne zakończenie scenariusza: Jeśli dany [użytkownik] nie istnieje lub został usunięty, to wyświetlony zostaje komunikat o błędzie.

Usunięcie użytkownika

Priorytet: wysoki

Aktorzy: administrator

Opis:

1. [Administrator] klika na odnośnik usunięcia użytkownika (dostępny m.in. na [ekran informacji o użytkowniku]).
2. Wyświetlona zostaje prośba o potwierdzenie usunięcia.
3. [System] usuwa [użytkownika] i wyświetlony zostaje komunikat o sukcesie.
4. Wyświetlony zostaje [ekran listy użytkowników].

Alternatywne zakończenie scenariusza: Jeśli [system] nie mógł usunąć użytkownika, to wyświetlony zostaje komunikat o błędzie oraz [ekran informacji o użytkowniku].

Dodatkowe informacje: Usunięcie [użytkownika] nie wiąże się z fizycznym skasowaniem rekordu z bazy danych, ale tylko z zaznaczeniem go jako *usuniętego*:

- Dalej dotyczyć go będą przypadki użycia: [(Wyświetlenie wyników zawodów)], [(Wyświetlenie historii startów)].
- Nie będzie się można na niego zalogować.
- Nie będzie można go przywrócić do poprzedniego stanu.
- Nie będą dotyczyć go inne przypadki użycia.
- Będzie można dodać do [systemu] nowe konto o takiej [nazwie] jak nazwa usuniętego.

Wyświetlenie zgłoszeń użytkownika

Priorytet: średni

Aktorzy: administrator

Opis:

1. Wyświetlony zostaje [ekran listy zgłoszeń] dla tego [użytkownika].

Alternatywne zakończenie scenariusza: Jeśli dany [użytkownik] nie istnieje lub został usunięty, to wyświetlony zostaje komunikat o błędzie oraz [ekran informacji o użytkowniku].

Wyświetlenie historii startów

Pre: [(Wyświetlenie użytkownika)] lub [(Wyświetlenie listy psów użytkownika)]

Priorytet: średni

Aktorzy: administrator

Opis:

1. Wyświetlony zostaje [ekran historii startów].
2. Historia zawiera pozycje z bazy danych wedle jednego z kryteriów:
 - (a) starty danego [użytkownika],
 - (b) starty danego [psa].
3. Przy każdym elemencie listy jest odnośnik do [ekranu informacji o użytkowniku].

2.2.9 Wyświetlenie listy startowej

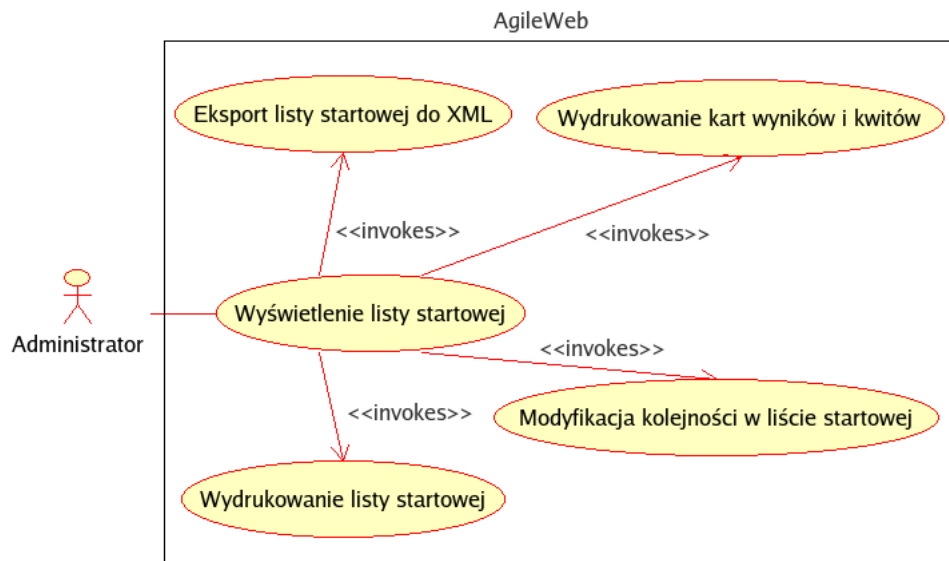
Eksport listy startowej do XML

Priorytet: wysoki

Aktorzy: administrator

Opis:

1. [System] wysyła do przeglądarki użytkownika nagłówek dokumentu MIME *application/xml*.



Rysunek 2.11: AgileWeb – przypadek użycia – Wyświetlenie listy startowej

2. [System] wysyła do przeglądarki plik XML z [listą startową].

Alternatywne zakończenie scenariusza: Jeśli [system] nie mógł wygenerować pliku XML, to wyświetlony zostaje komunikat o błędzie.

Wydrukowanie listy startowej, Wydrukowanie kart wyników i kwitów

Priorytet: średni

Aktorzy: administrator

Opis:

1. Wyświetlony zostaje [ekran listy startowej zawodów] lub [ekran kart wyników i kwitów] z arkuszem stylów CSS w wersji do druku. Na ekranie nie ma żadnych odnośników.

Modyfikacja kolejności w liście startowej

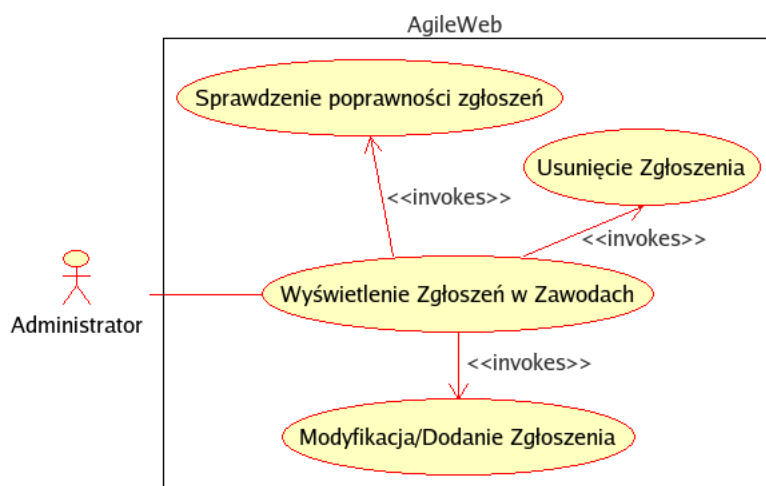
Priorytet: średni

Aktorzy: administrator

Opis:

1. Wyświetlony zostaje [ekran listy startowej zawodów], gdzie każdy element [listy startowej] posiada dwa odnośniki – przesunięcie w górę oraz w dół.
2. [Administrator] naciska jeden z odnośników (góra/dół) przy danym elemencie.
3. [System] dokonuje aktualizacji kolejności na [liście startowej].
4. Powrót do punktu 1.

2.2.10 Zarządzanie zgłoszeniami



Rysunek 2.12: AgileWeb – przypadek użycia – Zarządzanie zgłoszeniami

Wyświetlenie zgłoszeń w zawodach

Priorytet: wysoki

Aktorzy: administrator

Opis:

1. Wyświetlony zostaje [ekran listy zgłoszeń] ze [zgłoszeniami] wszystkich istniejących [użytkowników] w danych [zawodach].

Alternatywne zakończenie scenariusza: Jeśli nie ma żadnych [zgłoszeń] w [zawodach], to wyświetlona zostanie stosowny komunikat.

Sprawdzenie poprawności zgłoszeń

Priorytet: średni

Aktorzy: administrator

Opis:

1. Wyświetlony zostaje [ekran listy zgłoszeń].
2. Lista zawiera tylko [zgłoszenia] spełniające kryteria:
 - (a) [użytkownik] w przeszłości startował w [zawodach], a ma ustawiony parametr *debiutant*,
 - (b) [pies] został zgłoszony do kategorii wzrostowej nieprzeznaczonej dla jego wzrostu,
 - (c) [pies] ze stanem *weteran* nie został zgłoszony do kategorii wzrostowej o oczko niższej niż wynikałoby to z jego wzrostu.
3. Przy każdej pozycji na liście widnieje informacja o prawdopodobnej niezgodności.

2.3 Wymagania funkcjonalne – reguły biznesowe

Tabela 2.1: AgileWeb – wymagania funkcjonalne – konto i użytkownicy

Nr	Opis wymagania	Pr.
FR-U-1	Danymi uwierzytelniającymi przy logowaniu do konta użytkownika jest: [nazwa konta] oraz [hasło]. Podanie obydwu jest obligatoryjne.	W
FR-U-2	Jedno logowanie powinno być ważne minimum na czas trwania sesji w serwerze aplikacji.	W
FR-U-3	System umożliwia tylko [administratorowi] zmianę uprawnień innych [użytkowników].	W
FR-U-4	Uprawnienia [użytkowników] (wymienione w regule FR-U-3) dzielą się na: [użytkownik standardowy] oraz [administrator].	W
FR-U-5	System musi posiadać jedno nienaruszalne konto administratorskie (bez możliwości degradacji uprawnień). Konto to tworzone musi być w momencie instalacji systemu.	Ś
FR-U-6	Użytkownik niezalogowany w systemie (aktor: [gość]) może mieć dostęp do przypadków użycia: [rejestracja], [logowanie], [wyświetlenie listy zawodów], [wyświetlenie zawodów], [wyświetlenie wyników zawodów], [wyświetlenie list startowych].	W
FR-U-7	Każde konto niezakończony przez [administratora] musi być potwierdzone (aktywowane) przez [uczestnika]. W tym celu system musi wysłać list e-mail na adres podany w formularzu rejestracyjnym z odnośnikiem pozwalającym aktywować konto.	Ś
FR-U-8	Konto nieaktywowane posiada uprawnienia i przypadki użycia takie jak aktor [gość].	Ś
FR-U-9	System musi pozwolić na założenie wielu kont [użytkowników] z tym samym adresem e-mail.	Ś

Tabela 2.2: AgileWeb – wymagania funkcjonalne – uczestnicy i zgłoszenia do zawodów

Nr	Opis wymagania	Pr.
FR-Z-1	Jeden [użytkownik] może posiadać wiele [psów].	W
FR-Z-2	Jeden [użytkownik] może startować w danych [zawodach] z jednym [psem] lub więcej.	W
FR-Z-3	[Użytkownik] może modyfikować lub usunąć swoje [zgłoszenie] w [zawodach] do czasu ich rozpoczęcia.	W
FR-Z-4	Udział [użytkownika] w jednej z [konkurencji] w [zawodach] może wykluczyć jednoczesny udział w innej.	W
FR-Z-5	Administrator powinien móc określać wykluczenia pomiędzy [konkurencjami] z reguły FR-Z-4 .	Ś
FR-Z-6	Jedna [konkurencja] może mieć przypisaną jedną lub więcej [kategorię wzrostową].	W
FR-Z-7	Jedna [kategoria wzrostowa] może być przypisana do wielu [konkurencji].	W
FR-Z-8	Jeśli są [użytkownicy], którzy startują w danych [zawodach] kilka razy (z różnymi [psami]), to system powinien tak ustawiać [listy startowe], aby dany [użytkownik] nie miał dwóch biegów jeden po drugim.	Ś
FR-Z-9	Jeśli są [użytkownicy], którzy startują w danych [zawodach] kilka razy (z różnymi [psami]), to system powinien w miarę możliwości tak ustawiać [listy startowe], aby jeden [użytkownik] miał dłuższą przerwę pomiędzy biegami.	N
FR-Z-10	[Użytkownik] może usunąć lub zmienić swoje [zgłoszenie] dopóki nie rozpoczną się zawody.	Ś
FR-Z-11	[Użytkownik] może usunąć swojego psa w dowolnym momencie.	Ś

Tabela 2.3: AgileWeb – wymagania funkcjonalne – uczestnicy i zgłoszenia do zawodów c.d.

FR-Z-12	Usunięcie [psa] (zgodnie z regułą FR-Z-11) powoduje: - usunięcie [zgłoszeń] tego [użytkownika] z tym [psem]; - ustawienie stanu [psa] w bazie danych jako [usuniętego].	Ś
FR-Z-13	[Pies] ze stanem [usunięty] nie jest widoczny w konfiguracji konta użytkownika i w przypadkach użycia z tym związanych (tj. [modyfikacja zgłoszenia], [wyświetlenie swoich psów], [wybór psa]). Nie dotyczy go przypadek użycia [wyświetlenie psa]. Jednakże jego nazwa pojawi się w przypadkach użycia [wyświetlenie listy startowej] oraz [wyświetlenie wyników zawodów].	Ś
FR-Z-14	Jeden [użytkownik] nie może startować więcej niż 1 raz w tych samych [zawodach] z tym samym [psem] w tej tej samej [konkurencji].	W
FR-Z-15	Przy [zgłoszeniu] użytkownika po wyborze [konkurencji] proponowana jest odpowiednia [kategoria wzrostowa]. [Kategoria wzrostowa] jest dobierania bazując na [wysokości] [psa].	Ś
FR-Z-16	System powinien umożliwić tworzenie [konkurencji], w których [uczestnicy] zdobywają [punkty]. W takiej [konkurencji] o zwycięzcy decyduje największa liczba [punktów].	N
FR-Z-17	[Konkurencja] charakteryzowana powinna być przynajmniej przez pola: - [nazwa] (napis); - [dłuższy opis] (napis); - [rola punktów], tj. czy są to punkty karne czy zdobywane (wartość boolowska); - czy w konkurencji liczy się [czas] (wartość boolowska); - [długość toru] (liczba dziesiętna nieujemna).	Ś

Tabela 2.4: AgileWeb – wymagania funkcjonalne – listy startowe, wyniki końcowe i obsługa danych

Nr	Opis wymagania	Pr.
FR-D-1	Pozycje [listy startowej] o takiej samej parze [konkurencja]-[kategoria wzrostowa] muszą być w jednym ciągu.	W
FR-D-2	System musi eksportować [listy startowe] do pliku XML zgodnego z formatem oczekiwanym przez [AgileDesk].	W
FR-D-3	Postać drukowana [list startowych] powinna być czytelna.	W
FR-D-4	System musi importować [wyniki zawodów] z pliku XML zgodnego z formatem używanym przez [AgileDesk].	W
FR-D-5	Format plików XML (wymienione w regule FR-D-2 i FR-D-4) zostanie określony w osobnej specyfikacji.	W
FR-D-6	W importowanych [wynikach zawodów] system dopuszcza obecność [zawodników], którzy nie założyli konta w systemie.	Ś
FR-D-7	Lista [wyników zawodów] powinna być sortowana po liczbie [punktów] (rosnąco lub malejąco w zależności od typu konkurencji, patrz reguła FR-Z-16).	Ś

2.4 Wymagania pozafunkcjonalne

Tabela 2.5: AgileWeb – wymagania pozafunkcjonalne

Nr	Opis wymagania	Pr.
NFR-1	System powinien korzystać z transakcji przy dostępie do bazy danych.	Ś
NFR-2	Korzystanie z systemu odbywać się będzie poprzez przeglądarkę internetową.	W
NFR-3	System powinien zapewnić bezpieczeństwo przechowywanych danych użytkowników w tym informacji związanych z uwierzytelnieniem przy logowaniu do konta.	W

2.5 Ograniczenia środowiskowe

Tabela 2.6: AgileWeb – ograniczenia środowiskowe

Nr	Opis wymagania	Pr.
EC-1	System musi pracować w środowisku Apache, PHP 5 i MySQL.	W
EC-2	System powinien działać niezależnie od architektury systemu.	Ś

2.6 Diagram komponentów

Aplikacja została zaprojektowana zgodnie ze wzorcem MVC (por. rozdział 1.2.2). Wyróżnione zostały następujące komponenty:

Views – widoki, czyli warstwa prezentacji – elementy odpowiedzialne za kod wygląd interfejsu aplikacji (kod HTML). Poszczególne widoki ładowane są przez kontroler.

DataControllers – zebrane kilka kontrolerów odpowiedzialnych za podstawowe encje w systemie jak zawody, konkurencje, kategorie wzrostowe, psy użytkowników.

LoginController – kontroler obsługujący logowanie do aplikacji, przypomnienie hasła i aktywację konta.

UserController – komponent realizujący zakładanie i modyfikację konta jak i zarządzanie istniejącymi kontami przez administratora.

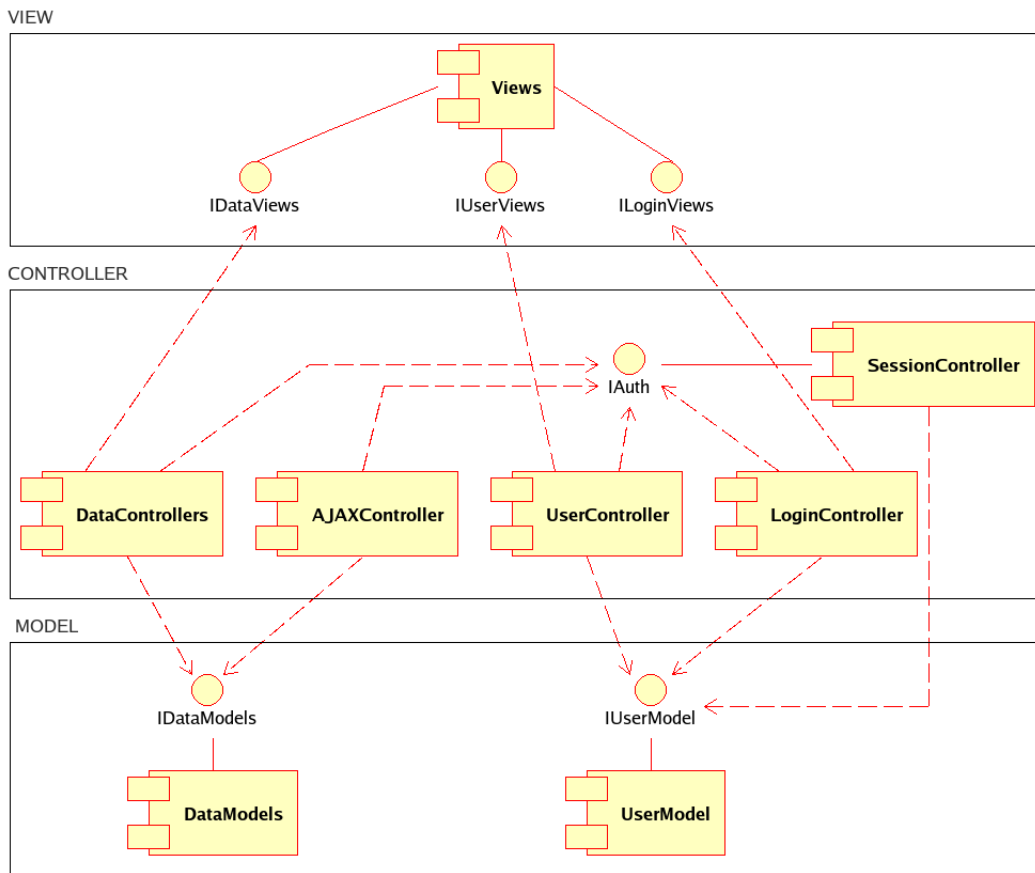
SessionController – komponent nie korzystający z żadnych widoków, właściwie pełniący rolę biblioteki. Obsługuje uwierzytelnienie i autoryzację użytkowników oraz sesję.

AJAXController – kontroler służący tylko do czynności związanych z zadaniami XMLHttpRequest (AJAX).

DataModels – zestaw modeli obsługujących podstawowe encje (zawody, konkurencje, kategorie wzrostowe itp.).

UserModel – model odpowiedzialny za konta użytkowników.

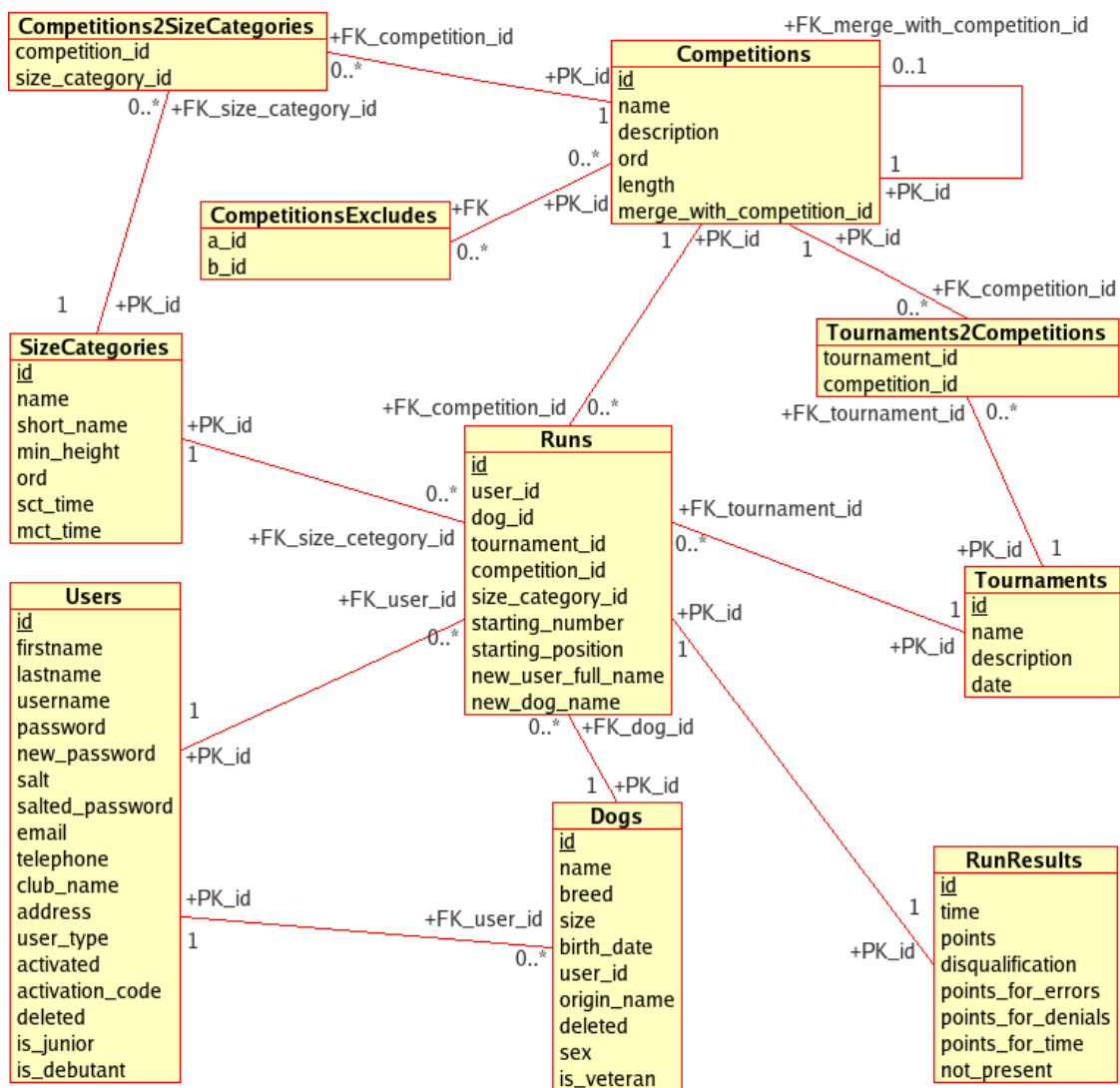
Architekturę aplikacji, podział na warstwy i komponenty prezentuje diagram 2.13.



Rysunek 2.13: AgileWeb – diagram komponentów i podział na warstwy

2.7 Diagram związków encji

Na diagramie 2.14 przedstawiony został projekt bazy danych. Na diagramie nie zostały umieszczone typy pól obiektów. Szczegóły implementacyjne bazy danych można ujrzyć w skrypcie instalacyjnym SQL (rozdział 3.1.3).



Rysunek 2.14: AgileWeb – diagram związków encji

Rozdział 3

Aplikacja AgileWeb

3.1 Instalacja

3.1.1 Wymagania

Jak wymienione zostało to w części *Ograniczenia środowiskowe* projektu AgileWeb (ograniczenie *EC-1* w tabeli 2.6), aplikacja instalowana musi być na serwerze WWW z obsługą PHP w wersji 5 lub wyższej. Serwerem WWW może być Apache [21] na platformie GNU/Linux lub Unix, ale równie dobrze wykorzystać można inny serwer wspierający używanie PHP (poprzez moduły współdzielone lub CGI/FastCGI).

AgileWeb bezpośrednio nie jest przywiązane do konkretnego DBMS (Database Management System), bo za obsługę bazy danych odpowiada framework CodeIgniter. Jednakże do poprawnej pracy DBMS musi obsługiwać transakcje, jak wymienione zostało w wymaganiu *NFR-1 z Wymagań pozafunkcjonalnych* projektu (tabela 2.5). Przykładami takich DBMS są:

- PostgreSQL [22];
- MySQL w wersji 5 lub nowszej (z użyciem tabel InnoDB) [20].

Dostęp do aplikacji AgileWeb odbywa się poprzez przeglądarkę WWW. Przeglądarka powinna obsługiwać XHTML 1.0 [23] oraz CSS 1.0 [24] oraz obiekt XMLHttpRequest [25] (poprzez język JavaScript). Powyższe wymagania spełniają nowe wersje przeglądarek: Opera, Mozilla Firefox, Konqueror, Safari oraz do pewnego stopnia Microsoft Internet Explorer w wersji 6.0 i 7.0. Na tych przeglądarkach aplikacja była testowana i sprawowała się poprawnie.

3.1.2 Instalacja plików

Proces instalacji zostanie zobrazowany na przykładzie uczelnianego serwera volt. Docelowym miejscem umiejscowienia plików będzie katalog `/home/stud/kozlowk3/WWW/agileweb/` dalej oznaczany przez `~/`, widoczny w internecie jako `http://iem.pw.edu.pl/~kozlowk3/agileweb`.

Pierwszym etapem instalacji AgileWeb jest wgranie plików CodeIgnitera. Jego instalacja jest bardzo prosta – szczegółowo opisuje ją też podręcznik użytkownika [7].

1. Pobranie ze strony CodeIgnitera [5] aktualnej paczki z frameworkiem.
2. Rozpakowanie archiwum do katalogu na serwerze WWW, w którym będzie umieszczona cała aplikacja AgileWeb:

```
unzip CodeIgniter_1.5.4.zip
mv CodeIgniter_1.5.4/* ~/
```

3. Edycja pliku `~/system/application/config/config.php` i ustawienie w nim:
 - `$config['base_url']` – na adres URL do aplikacji, czyli miejsca z plikiem `index.php` (z końcowym slashem), np.: `http://iem.pw.edu.pl/~kozlowk3/agileweb/`;
 - `$config['sess_use_database']` – na wartość `TRUE`;
 - `$config['sess_match_ip']` – na wartość `TRUE`;
 - `$config['cookie_domain']` – na nazwę domeny, w której mają być tworzone ciasteczka (opcjonalne), np. `iem.pw.edu.pl`;
 - `$config['cookie_path']` – na ścieżkę w adresie URL dla ustawianych ciasteczek (opcjonalne), np. `/~kozlowk3/agileweb/`;
4. Edycja pliku `~/system/application/config/agilewebconstants.php` i ustawienie w nim:
 - `FROM_MAIL` – na adres email, z którego mają być wysyłane maile, np.: `webmaster@iem.pw.edu.pl`;
 - `WEBSITE_ADMIN` – na dane kontaktowe do administratora;
5. Edycja pliku `~/system/application/config/database.php` i ustawienie w nim połączenia z bazą danych:
 - `$db['default']['hostname']` – nazwa hosta bazy danych (np. `localhost`);

- `$db['default']['username']` – nazwa użytkownika;
- `$db['default']['password']` – hasło;
- `$db['default']['database']` – nazwa bazy danych;
- `$db['default']['dbprefix']` – prefiks nazwy każdej z tabel, skrypt instalacyjny SQL dołączony do aplikacji AgileWeb wymaga by ten prefiks brzmiał `agileweb_` (zwróć uwagę na końcowe podkreślenie);

Kolejnym etapem jest wgranie biblioteki XAJAX w wersji 0.2.x. Z pobranego ze strony domowej projektu [9] archiwum należy wypakować pliki i:

1. katalog `xajax_js` wraz z zawartością umieścić w korzeniu aplikacji AgileWeb (tj. obok pliku `index.php` z CodeIgniter);
2. plik `xajax.inc.php` umieścić w `~/system/application/libraries` zmieniając nazwę na `xajax.php`;
3. pliki `xajaxCompress.php` oraz `xajaxResponse.inc.php` umieścić w `~/system/application/libraries` (bez zmiany nazwy);

3.1.3 Baza danych

Obok kodu źródłowego dołączonego na nośniku do pracy znaleźć można plik SQL `agileweb_install.sql` instalujący wymagane tabele w bazie danych. Plik przeznaczony jest dla MySQL i wymaga wsparcia dla tabel InnoDB. Przykładowo korzystając z konsolowego klienta `mysql` polecenie utworzenia wymaganej struktury bazy danych może wyglądać następująco:

```
mysql -p -u nazwa_uzytkownika \  
    nazwa_bazy_danych < agileweb_install.sql
```

Skrypt `agileweb_install.sql` utworzy tabele w bazie o nazwach zaczynających się od `agileweb_`, stąd ważne jest ustawienie zmiennej `dbprefix` w konfiguracji dostępu do bazy danych w CodeIgniter. Skrypt doda też konto podstawowego administratora systemu – użytkownika `admin` z hasłem `admin`.

3.2 Instrukcja użytkownika

3.2.1 Rozpoczęcie pracy – konto

Użytkownik niezalogowany ma ograniczony dostęp do aplikacji. Może tylko ujrzeć listę zawodów oraz ich wyniki lub listy startowe. Dalsza funkcjonalność zostaje udostępniona po utworzeniu konta i zalogowaniu się na nie.

[Strona główna](#) / utworzenie nowego konta

Utworzenie nowego konta

Pola obowiązkowe:

Nazwa użytkownika:

Hasło:

Powtórz hasło:

Adres e-mail:

Imię:

Nazwisko:

Pola nieobowiązkowe:

Telefon:

Ulica: (ulica, numer domu, numer mieszkania)

Kod pocztowy:

Miasto:

Junior: (nie ukończył 18 lat do dnia zawodów)

Debiutant:

Nazwa klubu:

Zalóż konto

Rysunek 3.1: AgileWeb – ekran rejestracji w systemie

Po wypełnieniu formularza rejestracyjnego (rys. 3.1) na podany adres e-mail zostanie wysłany list potwierdzający założenie konta. W liście znajdować się będzie adres URL, po którego odwiedzinach konto zostanie aktywowane. Do czasu aktywacji konto ma te same uprawnienia, co użytkownik niezalogowany – gość. Użytkownik identyfikowany jest w systemie poprzez nazwę konta oraz hasło.

Jeśli użytkownik zapomni hasło do swojego konta, może skorzystać z opcji jego przypomnienia. Będzie jednakże musiał podać prawidłową nazwę konta i adres e-mail z nim związany. Jeśli obydwie dane poda poprawne, to na ten adres zostanie wysłane nowe, tymczasowe hasło, na które będzie mógł się zalogować. Do czasu użycia tego tymczasowego hasła, jego główne nie zostanie zmienione. Po zalogowaniu się użytkownik otrzymuje dostęp do zarządzania swoim kontem w ramach aplikacji (rys. 3.2). Poza zmianą danych osobowych, hasła czy adresu e-mail powinien dodać w nim psy, z którymi zamierza startować w zawodach.

AgileWeb

[Strona główna](#) [O aplikacji/Pomoc](#) [Zawody](#) [Konto](#) [Wyloguj się](#)

[Strona główna](#) / [test test](#)

test test - konto

Nazwa użytkownika: **test test**
 Adres e-mail: **tester**
 Junior: **kozik1@o2.pl**
 Debiutant: **nie**
 Telefon: **nie**
 Adres:
 Nazwa klubu:

- ◆ [Modyfikacja konta](#)
- ◆ [Dodaj nowego psa](#)

Twoje psy

Imię	Rasa	Płeć	Wysokość	Data urodzenia	Imię rodowodowe	Weteran
Reks	jamnik	pies	20 cm	2002-10-10		zmień / usuń
Azor	Kundel	pies	40 cm	2000-05-11		tak: zmień / usuń

Rysunek 3.2: AgileWeb – ekran konta użytkownika

3.2.2 Udział w zawodach

Posiadając konto w systemie użytkownik może zgłosić udział w zawodach do dniach ich rozpoczęcia. Przy każdym czynnych zawodach widnieć będzie odnośnik *zgłoś uczestnictwo* prowadzący do formularza dodania lub modyfikacji swojego zgłoszenia udziału.

[Strona główna](#) / [zawody - testowe w test](#) / [modyfikacja zgłoszenia](#)

Zgłoszenie do zawodów testowe w test

Zgłoszenie nr 1:

Pies:

Konkurencja:

Kategoria wzrostowa:

[Usuń to zgłoszenie](#)

Rysunek 3.3: AgileWeb – ekran modyfikacji zgłoszenia w zawodach

Ekran z formularzem (rys. 3.3) zawiera wszystkie zgłoszenia danego użytkownika w tych zawodach. Przy istniejących zgłoszeniach umieszczony jest odnośnik *Usuń to zgłoszenie* w celu jego anulowania. W każdym zgłoszeniu

użytkownik musi wybrać jednego ze swoich psów, konkurencję i kategorię wzrostową. Dostępne dla danego psa konkurencje oraz kategorie wzrostowe są uzależnione od samej konfiguracji zawodów ustalanej przez administratora oraz ewentualnie od wcześniejszych zgłoszeń w tych zawodach. Aplikacja, poprzez AJAX, reaguje na wybór użytkownika bez konieczności ponownego ładowania formularza:

- po wyborze psa zostanie uaktualniona lista wyboru konkurencji (bazując na ewentualnych wykluczeniach pomiędzy jednoczesnym udziałem w kilku konkurencjach);
- po wyborze konkurencji zostanie uaktualniona lista dostępnych kategorii wzrostowych w tej konkurencji oraz pojawi się nazwa sugerowanej kategorii wzrostowej bazując na wysokości danego psa.

Po uzupełnieniu pól formularza (psa, konkurencji i kategorii wzrostowej), aby zgłoszenie zostało zachowane, użytkownik powinien nacisnąć *Zapisz zmiany*. Użytkownik może modyfikować (w tym i usuwać) zgłoszenia w zawodach do czasu ich rozpoczęcia.

3.2.3 Zarządzanie zawodami (administrator)

Zarządzanie zawodami możliwe jest tylko z konta o uprawnieniach administratora. Wszystkie elementy dalej opisane znaleźć można pod zakładką *Zawody* z menu aplikacji.

Strona główna / lista zawodów / zmiana konkurencji

Zmiana konkurencji

Zmień konkurencje:

Nazwa konkurencji	Opis	Kolejność	Dług. toru	Połącz z
Agility		1	0	Jumping <input type="button" value="usuń"/>
Jumping		2	0	Agility <input type="button" value="usuń"/>
Youngsters	Dla młodych psów	3	0	brak <input type="button" value="usuń"/>

Dodaj nową konkurencję:

Nazwa konkurencji	Opis	Kolejność	Dług. toru	Połącz z
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	brak <input type="button" value="usuń"/>

Rysunek 3.4: AgileWeb – ekran modyfikacji konkurencji

Administrator powinien wpieryw dodać odpowiednią ilość konkurencji (rys 3.4), które będą przypisywane poszczególnym zawodom. Samo przypisanie

konkurencji do zawodów odbywa się przy dodawaniu nowych lub modyfikacji istniejących zawodów. Tworzenie konkurencji łącznych (będących zasadniczo tylko częścią wspólną wyników biegów z obydwu) odbywa się poprzez ustawienie pola *Połącz z* na nazwę drugiej konkurencji. Jedna konkurencja łączona może składać się tylko z dwóch konkurencji.

Administrator może ustawić wykluczenia pomiędzy konkurencjami. Wykluczenie polega na zabronieniu zawodnikowi startu w danej konkurencji, jeśli już zgłosił swoje uczestnictwo do innej (wykluczającej).

[Strona główna](#) / [lista zawodów](#) / [łączenie kategorii wzrostowych i konkurencji](#)

Łączenie kategorii wzrostowych i konkurencji

Obecne połączenia:

Konkurencja	Kategoria wzrostowa	
Agility	Small	usuń
Agility	Medium	usuń
Agility	Large	usuń

Dodaj nowe połączenie:

Konkurencja	Kategoria wzrostowa
Agility	Small

Rysunek 3.5: AgileWeb – ekran łączenia kategorii wzrostowych i konkurencji

Podobnie funkcjonuje ekran modyfikacji kategorii wzrostowych. Administrator powinien utworzyć wszystkie wymagane kategorie wzrostowe zwracając szczególną uwagę na pola:

minimalna wysokość psa – używane przy sugerowaniu użytkownicy kategorii wzrostowej przy zgłoszeniu oraz przy sprawdzaniu poprawności zgłoszeń;

czasy SCT i MCT – mające udział w liczeniu punktów karnych za czas i ewentualnej dyskwalifikacji;

Po zakończonym przygotowywaniu konkurencji i kategorii wzrostowych administrator powinien je powiązać, tzn. przypisać poszczególne kategorie wzrostowe do konkurencji. Każda konkurencja musi mieć przynajmniej jedną kategorię wzrostową.

3.2.4 Lista startowa (administrator)

Początkowo lista startowa (rys. 3.6) zawiera biegi w kolejności zgłoszeń, a zawodnicy nie mają numerów startowych.

Strona główna / zawody - testowe w test / lista startowa

Lista startowa zawodów testowe w test

Skocz do:

- ◆ [Agility](#)
- ◆ [Jumping](#)
- ◆ [Youngsters](#)
- ◆ [Zerówka](#)

Konkurencja: Agility

Kol.	Nr start.	Przewodnik	Pies	Kat. wzr.	
Small					
1	1	Adminowicz Pierwszy	Azor	Small	usuń / zmień / w dół / w górę
2	6	AAA	BBB	Small	usuń / zmień / w dół / w górę
3	0	test test	Reks	Small	usuń / zmień / w dół / w górę
Medium					
1	2	Jan Kowalski	Burek	Medium	usuń / zmień / w dół / w górę
2	3	1 4	Reks	Medium	usuń / zmień / w dół / w górę
3	4	Adminowicz Pierwszy	Burek	Medium	usuń / zmień / w dół / w górę
Large					

Rysunek 3.6: AgileWeb – ekran listy startowej

System umożliwia automatyczne sortowanie listy w celu zapewnienia odpowiedniego odstępu przed startem tych samych przewodników (reguła *FR-Z-8* z wymagań funkcjonalnych, tabela 2.2). Przesortowanie listy nada również każdemu zawodnikowi numer startowy (o ile go nie ma).

Jeśli istnieje potrzeba ręcznej poprawy kolejności startów, administrator może dowolnie przesuwając biegi poprzez naciśnięcie “*w górę*” lub “*w dół*”. Jeśli po ostatnim sortowaniu listy zostały dodane zgłoszenia nowych zawodników, to należy nadać im numery startowe – albo poprzez ponowne automatyczne przesortowanie listy albo poprzez odnośnik “*Nadaj numery startowe*”. Uwaga – nie dotyczy to kolejnych zgłoszeń zawodnika już uczestniczącego w zawodach, gdyż system dba o to, aby jedna para przewodnik–pies miała w zawodach ten sam numer startowy w każdym biegu.

Po zakończonych czynnościach modyfikacji listy startowej administrator może sprawdzić poprawność zgłoszeń (szczegółowo omówione w wymaganiu *Sprawdzenie poprawności zgłoszeń*, rozdział 2.2.10) lub wyeksportować ją do pliku XML dla aplikacji AgileDesk.

Zarówno ekran listy startowej jak i jego pochodne (tj. ekran listy zgłoszeń, ekran druków dla przebiegów, ekran listy z miejscami na wyniki) posiadają swoje specjalne wersje do druku z alternatywnymi arkuszami styli CSS.

3.2.5 Wyniki zawodów (administrator)

Po zakończonych zawodach administrator musi zaimportować wyniki zawodów z wcześniej wygenerowanego pliku przez aplikację AgileDesk.

Konkurencja: Jumping								
Pies	Przewodnik	Numer startowy	Czas	Punkty karne				
Small, SCT: 20,00s, MCT: 40,00s				błędy	odmowy	za czas	łącznie	
-- brak --								
Medium, SCT: 15,00s, MCT: 30,00s				błędy	odmowy	za czas	łącznie	
1	Azor	Adminowicz Pierwszy	1	8.6	1	0	0	1
	Burek	Jan Kowalski	2					Dis.
Large, SCT: 10,00s, MCT: 20,00s				błędy	odmowy	za czas	łącznie	
-- brak --								

Konkurencja: Zerówka								
Pies	Przewodnik	Numer startowy	Czas	Punkty karne				
Small, SCT: 20,00s, MCT: 40,00s				błędy	odmowy	za czas	łącznie	
-- brak --								

Rysunek 3.7: AgileWeb – ekran wyników zawodów

Operacja sprowadza się tylko do wyboru pliku XML i przesłania go do aplikacji AgileWeb. Wcześniejsze wyniki danych zawodów zostaną usunięte przed importem nowych. W importowanych wynikach mogą znaleźć się biegi zawodników nieposiadających konta w systemie – pojawią się oni na liście wyników zawodów oraz zostanie uaktualniona lista startowa.

Wyniki zawodów (rys. 3.7) dostępne są przez cały czas od ich wprowadzenia. Ekran ten posiada posiadają swoje specjalne wersje do druku z alternatywnymi arkuszami styli CSS tak, aby drukowana postać była czytelna.

Rozdział 4

Projekt aplikacji AgileDesk

4.1 Opis systemu

4.1.1 Wizja systemu

System ma na celu rozwiązanie problemów związanych z obsługą przebiegu zawodów sportowych. Dzięki wykorzystaniu systemu:

- będzie można wygodnie wprowadzać ostatnie poprawki do list startowych bezpośrednio podczas zawodów;
- zostanie usprawniony i znormalizowany proces rejestracji wyników biegów zawodników;
- uproszczone zostanie przetwarzanie wyników z zawodów wraz z ich drukowaniem.

4.1.2 Funkcjonalne wymagania odnośnie systemu

Obsługa list startowych – import list startowych zawodników z pliku XML, wyświetlanie ich w postaci przeznaczonej do druku i na ekranie monitora/projektora.

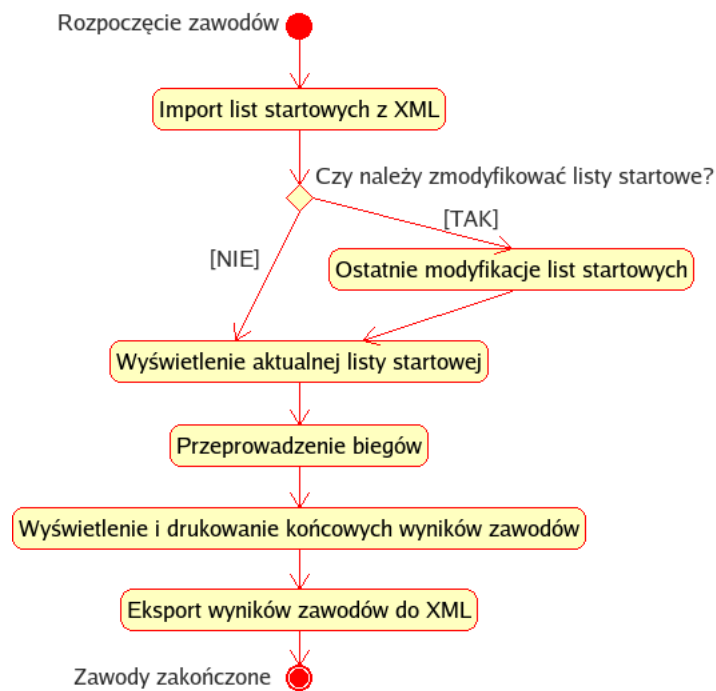
Modyfikacje w czasie zawodów – umożliwienie wprowadzania modyfikacji do list startowych czy danych użytkowników w trakcie trwania zawodów.

Zapisywanie pomiarów biegów – przyjmowanie i przechowywanie wyników biegów zawodników i generowanie końcowej klasyfikacji z zawodów.

Połączenie z AgileWeb – synchronizacja wyników zawodów z systemem AgileWeb.

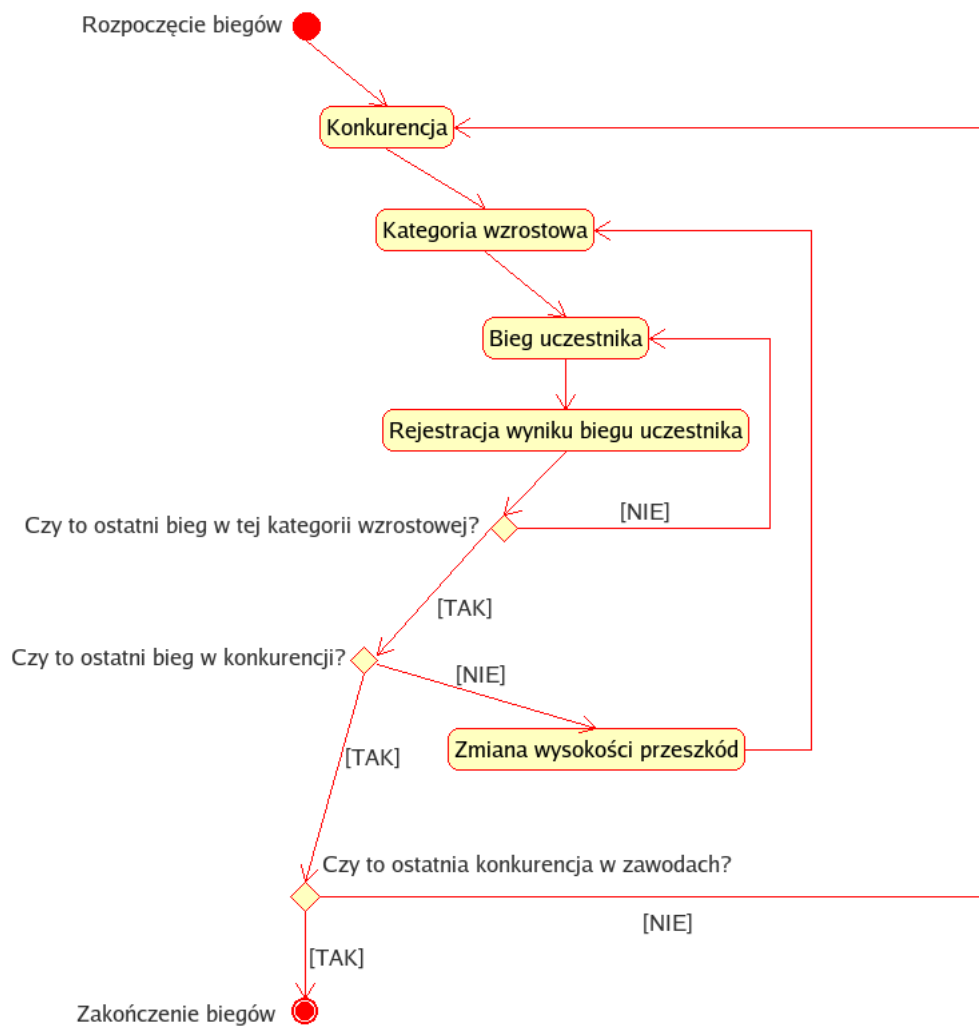
4.1.3 Opis procesów biznesowych

Aplikacja AgileDesk obejmuje jedynie sam przebieg faktycznych zawodów. Ich przebieg, z podziałem na główne czynności, przedstawia diagram 4.1.



Rysunek 4.1: Organizacja zawodów – schemat

Akcja *Przeprowadzenie biegów* została z kolei opisana dokładniej na diagramie 4.2. Zawody dzielą się na konkurencje, te na kategorie wzrostowe, a do tych przyporządkowane są poszczególne biegi. Stąd też na tym diagramie trzy pętle.



Rysunek 4.2: Biegi w zawodach – schemat

4.2 Wymagania funkcjonalne – przypadki użycia i scenariusze

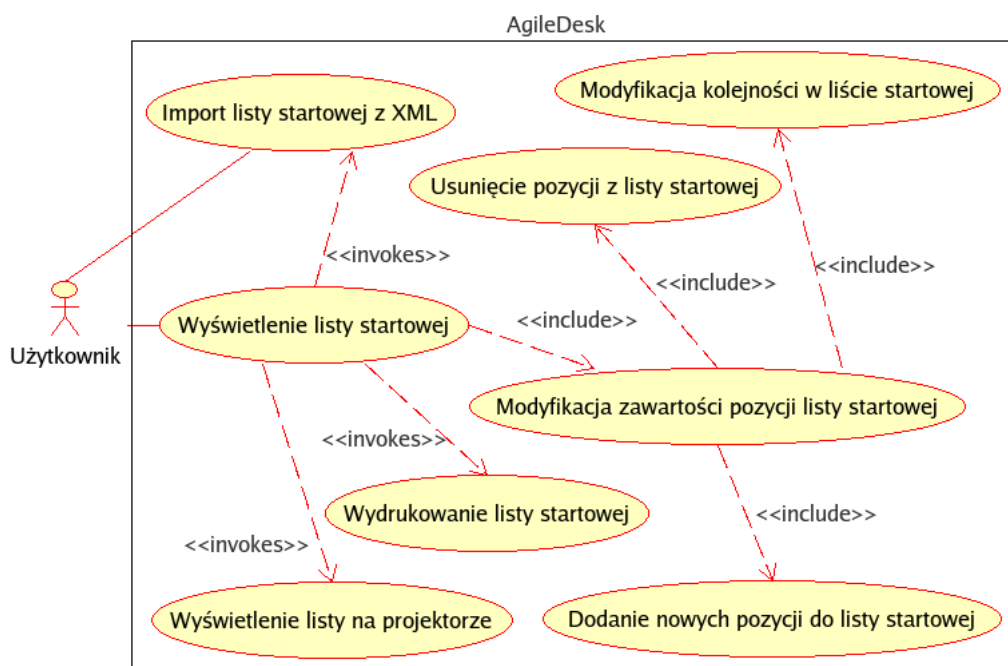
4.2.1 Aktorzy

- **Użytkownik** – Zdefiniowany jest tylko jeden aktor – użytkownik.



Rysunek 4.3: AgileDesk – Aktorzy w systemie

4.2.2 Lista startowa



Rysunek 4.4: AgileDesk – przypadek użycia – Lista startowa

Import listy startowej z XML

Priorytet: wysoki

Opis:

1. Wyświetlone zostaje [okno wyboru pliku] XML z listą startową. W oknie wyświetlone są domyślnie tylko pliki XML (filtr ten można zmienić na wyświetlanie wszystkich plików).
2. Użytkownik wybiera plik.
3. System importuje listę startową i wyświetla okienko z informacją o pomyślnym wykonaniu operacji.

Alternatywne zakończenie scenariusza: Jeśli wskazany plik nie był w prawidłowym formacie listy startowej XML lub jego import nie powiódł się, to wyświetlony zostaje komunikat o błędzie.

Wyświetlenie listy startowej

Priorytet: wysoki

Opis:

1. Wyświetlone zostaje [okno listy startowej] na podstawie aktualnego stanu aplikacji.
2. Lista składa się z wierszy – każdy przedstawiający start jednego zawodnika w danej kategorii wzrostowej i konkurencji.
3. Użytkownik może modyfikować kolejność tych wierszy poprzez zmianę odpowiedniego pola tekstowego.
4. Użytkownik może usunąć wybrany wiersz poprzez naciśnięcie na odpowiedni przycisk.
5. Użytkownik może modyfikować pola tekstowe w wierszu tym samym modyfikując dane tylko w nim. Modyfikacje nazwiska zawodnika lub imienia psa w danym wierszu nie oznaczają jego modyfikacji w pozostałych.

Alternatywne zakończenie scenariusza: Jeśli nie została wczytana żadna lista startowa, stan aplikacji jest pusty lub użytkownik próbował zmienić pole na niedozwoloną wartość (np. na pustą), to wyświetlony zostaje komunikat o błędzie.

Wyświetlenie listy na projektorze, Wydrukowanie listy startowej

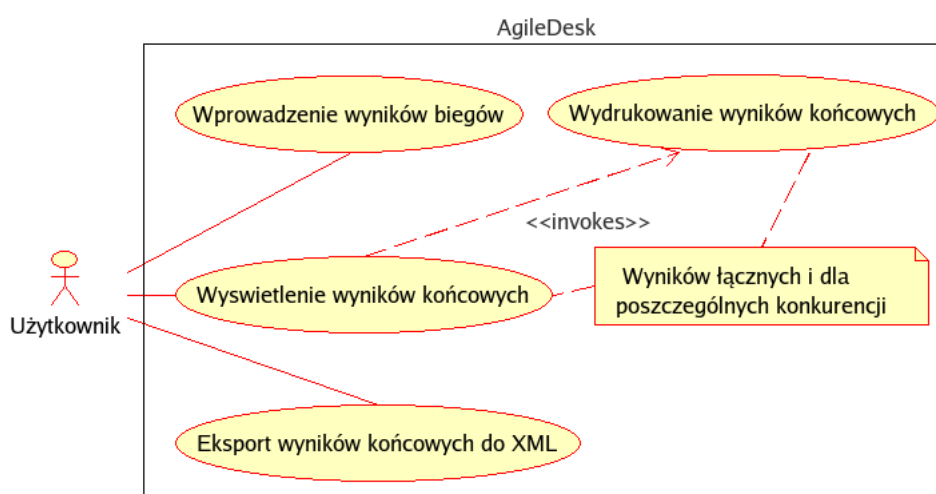
Priorytet: średni (dla wyświetlenia), niski (dla wydrukowania)

Opis:

1. Wyświetlone zostaje [okno listy podglądu startowej].
2. Okno nie umożliwia modyfikacji żadnych danych – stanowi tylko czytelniejszą formę przedstawienia listy.

Alternatywne zakończenie scenariusza: Jak w przypadku *Wyświetlenie listy startowej*.

4.2.3 Wyniki biegów



Rysunek 4.5: AgileDesk – przypadek użycia – Wyniki biegów

Wprowadzenie wyników biegów

Priorytet: wysoki

Opis:

1. Wyświetlone zostaje [okno wprowadzania wyników biegów] zawierające w wierszach poszczególne biegi zawodników (patrz przypadek użycia [(Wyświetlenie listy startowej)]) wraz z polami.
2. Użytkownik może wypełnić wyniki składające się na dany bieg, tj.:
 - liczba punktów karnych za czas,
 - liczba punktów karnych za błędy,
 - liczba punktów karnych za odmowy,

- czas.
3. Punkty karne za odmowy i błędy mogą być wprowadzane na zasadzie wypełnienia pola tekstowego lub poprzez naciśnięcie na przycisk inkrementacji o zadaną wartość.
 4. W każdym wierszu jest przycisk uaktywniający pomiar czasu biegu. Pierwsze jego wciśnięcie włącza pomiar, drugie wyłącza i powoduje zapisanie zmierzonej wartości w polu [czas] dla danego biegu (nadpisując obecną tam wartość). Pomiar czasu może być liczony w ten sposób dla wielu biegów niezależnie.
 5. Jeśli pole [czas] jest wypełnione, to użytkownik może nacisnąć przycisk *przelicz punkty* powodujący obliczenie ilości punktów karnych związanych z przekroczeniem wartości SCT ([średni czas przebiegu]). W przypadku, gdy [czas] przekracza wartość MCT ([maksymalny czas przebiegu]), następuje dyskwalifikacja zawodnika w tym biegu. Kategoria wzrostowa, której dotyczy dany bieg, musi mieć wartości MCT i SCT zdefiniowane.
 6. Użytkownik może zdyskwalifikować zawodnika poprzez naciśnięcie na przycisk *dyskwalifikacja*.
 7. Wszystkie wartości dla danego biegu (włącznie z dyskwalifikacją) mogą zostać wyczyszczone poprzez naciśnięcie na przycisk *reset*.

Eksport wyników końcowych do XML

Priorytet: wysoki

Opis:

1. Wyświetlone zostaje [okno wyboru pliku] XML dla wyników zawodów. W oknie wyświetlone są domyślnie tylko pliki XML (filtr ten można zmienić na wyświetlanie wszystkich plików).
2. Użytkownik może wprowadzić nazwę nowego pliku lub wybrać jeden z dostępnych na dysku.
3. System eksportuje wyniki i wyświetla komunikat o pomyślnym wykonaniu operacji.

Alternatywne zakończenie scenariusza: W przypadku napotkania błędu podczas eksportu lub braku wyników zawodów wyświetlony zostaje komunikat o błędzie.

Wyświetlenie wyników końcowych, Wydrukowanie wyników końcowych

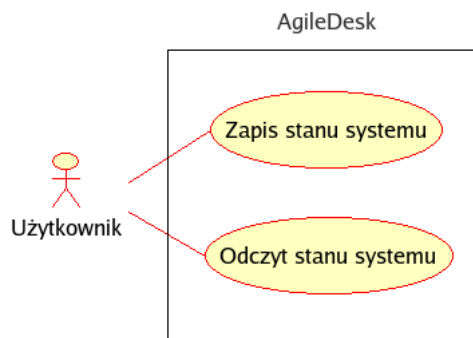
Priorytet: średni (dla wyświetlenia), niski (dla wydrukowania)

Opis:

1. Wyświetlone zostaje [okno wyników zawodów].

Alternatywne zakończenie scenariusza: W przypadku braku wyników zawodów wyświetlony zostaje komunikat o błędzie.

4.2.4 Obsługa stanu



Rysunek 4.6: AgileDesk – przypadek użycia – Obsługa stanu

Zapis stanu systemu

Priorytet: wysoki

Opis:

1. Wyświetlone zostaje [okno wyboru pliku], do którego zostanie zapisany stan systemu. W oknie wyświetlone są domyślnie tylko pliki XML (filtr ten można zmienić na wyświetlanie wszystkich plików).
2. Użytkownik może wprowadzić nazwę nowego pliku lub wybrać jeden z dostępnych na dysku.
3. System zapisuje obecny stan i wyświetla komunikat o pomyślnym wykonaniu operacji.

Alternatywne zakończenie scenariusza: W przypadku napotkania błędu podczas zapisywania do pliku wyświetlony zostaje komunikat o błędzie.

Odczyt stanu systemu

Priorytet: wysoki

Opis:

1. Wyświetlone zostaje [okno wyboru pliku], z którego zostanie utworzony stan systemu. W oknie wyświetlone są domyślnie tylko pliki XML (filtr ten można zmienić na wyświetlanie wszystkich plików).
2. System odczytuje plik i ustawia prawidłowy stan.

Alternatywne zakończenie scenariusza: W przypadku napotkania błędu podczas odczytu pliku (np. jego nieprawidłowego formatu) wyświetlony zostaje komunikat o błędzie.

4.3 Wymagania funkcjonalne (reguły biznesowe)

Tabela 4.1: AgileDesk – wymagania funkcjonalne – lista startowa

Nr	Opis wymagania	Pr.
FR-L-1	System musi importować [listę startową] z pliku XML zgodnego z formatem używanym przez [AgileWeb].	W
FR-L-2	System powinien umożliwić ręczną modyfikację kolejności pozycji w [liście startowej].	W
FR-L-3	System powinien umożliwić usunięcie, dodanie lub modyfikację pozycji z [listy startowej].	W
FR-L-4	Przy dodaniu nowej pozycji do [listy startowej] (z reguły FR-L-3) system przechowa wprowadzone dane jako nowego [zawodnika].	W
FR-L-5	System powinien umożliwić wydrukowanie zmodyfikowanej [listy startowej] w czytelnej postaci.	N
FR-L-6	System powinien umożliwić wyświetlenie zmodyfikowanej [listy startowej] w czytelnej postaci (np. na projektorze).	Ś
FR-L-7	Obowiązkowymi polami w [liście startowej] są: <ul style="list-style-type: none"> - [imię i nazwisko użytkownika] (napis) - [nazwa psa] (napis) - [pozycja na liście startowej] (liczba całkowita nieujemna) - [numer startowy] (liczba całkowita nieujemna) - [kategoria wzrostowa] (napis) - [konkurencja] (napis). 	W
FR-L-8	System powinien umożliwić sortowanie wyświetlanej [listy startowej] (z reguły FR-L-6) po następujących polach: [nazwisko użytkownika], [numer startowy].	N
FR-L-9	System powinien umożliwić wprowadzanie dodatkowej informacji o [zawodnikach] takich jak: [obecność na zawodach] (wartość boolowska).	N
FR-L-10	Dodawane przez użytkownika pozycje do [listy startowej] są umieszczane na jej końcu.	W

Tabela 4.2: AgileDesk – wymagania funkcjonalne – wyniki biegów i zawodów

Nr	Opis wymagania	Pr.
FR-W-1	System powinien umożliwić wydrukowanie [wyników końcowych] w czytelnej postaci.	N
FR-W-2	System powinien umożliwić wyświetlenie [wyników końcowych] w czytelnej postaci (np. na projektorze).	Ś
FR-W-3	System musi wyeksportować aktualne [wyniki zawodów] do pliku XML zgodnego z formatem używanym przez [AgileWeb].	W
FR-W-4	Wyświetlenie [wyników końcowych] (wymienionych w regułach FR-W-1 i FR-W-2) obejmuje: wyniki łączne (dla konkurencji łączonych) oraz wyniki dla poszczególnych konkurencji.	Ś
FR-W-5	Na [wynik] każdego z biegów składa się: - ilość [punktów] karnych za czas (liczba dziesiętna nieujemna); - ilość [punktów] karnych za błędy (liczba całkowita nieujemna); - ilość [punktów] karnych za odmowy (liczba całkowita nieujemna); - [czas] (czas z dokładnością do setnej części sekundy); - [dyskwalifikacja] (wartość boolowska).	W
FR-W-6	System ma umożliwić użytkownikowi ręczne wprowadzenie każdego elementu [wyniku] biegu (wymienionych w regule FR-W-5) dla każdej pozycji w [liście startowej].	W
FR-W-7	System powinien umożliwić przeprowadzenie pomiaru [czasu] dla każdego biegu. Rozpoczęcie i zakończenie pomiaru odbywa się poprzez odpowiednią akcję użytkownika. Po zakończeniu pomiaru pole z [czasem] biegu powinno zostać wypełnione zmierzonym czasem.	W
FR-W-8	Dokładność pomiaru (z reguły FR-W-7) określają możliwości sprzętowe i systemowe komputera z uruchomionym systemem.	W
FR-W-9	Niezależnie od wykonania pomiaru [czasu] biegu (z reguły FR-W-7) użytkownik powinien móc wprowadzić ręczne poprawki.	W

Tabela 4.3: AgileDesk – wymagania funkcjonalne – wyniki biegów i zawodów c.d.

Nr	Opis wymagania	Pr.
FR-W-10	System powinien rozróżnić [konkurencje], w których [uczestnicy] zdobywają [punkty] za pokonywanie przeszkód, od [konkurencji] z punktami karnymi za popełnione błędy lub odmowy.	N
FR-W-11	Dla zwykłych [konkurencji] niewprowadzenie [czasu] biegu oznacza brak startu zawodnika w tym biegu. Dla [konkurencji] ze zdobywaniem [punktów] (patrz reguła FR-W-10) niewprowadzenie [czasu] nie niesie żadnych dalszych konsekwencji.	Ś
FR-W-12	Niewprowadzenie [punktów karnych] oznacza wartość 0.	W
FR-W-13	Wprowadzenie wartości [dyskwalifikacja] dla biegu automatycznie wyłącza pola tekstowe do wpisania pozostałych parametrów biegu.	W
FR-W-14	System powinien umożliwić modyfikację [wyników] dowolnego biegu.	W
FR-W-15	Drukowane lub wyświetlane [wyniki końcowe] (patrz reguły FR-W-1 i FR-W-2) powinny być posortowane po ilości punktów. Rosnąco dla zwykłej [konkurencji] i malejąco dla [konkurencji] ze zdobywaniem [punktów].	W

Tabela 4.4: AgileDesk – wymagania funkcjonalne – obsługa danych

Nr	Opis wymagania	Pr.
FR-D-1	System musi mieć możliwość zachowania i odtworzenia swojego stanu.	W
FR-D-2	Stanem aplikacji wymienionym w regule FR-D-1 jest aktualna [lista startowa], informacje dodatkowe wprowadzone w regułach FR-L-4 i FR-L-9 oraz aktualne [wyniki biegów].	W
FR-D-3	Import [listy startowej] (z reguły FR-1) wpierw czyści (resetuje) stan systemu.	W
FR-D-4	Format plików XML (wymienione w regule FR-L-1 i FR-W-3) zostanie określony w osobnej specyfikacji.	W

4.4 Wymagania pozafunkcjonalne

Tabela 4.5: AgileDesk – wymagania pozafunkcjonalne

Nr	Opis wymagania	Pr.
NFR-1	System powinien mieć interfejs graficzny.	W
NFR-2	System powinien móc być obsługiwany przez klawiaturę lub myszkę.	W
NFR-3	System powinien zapewnić bezpieczeństwo przechowywanych danych użytkowników oraz wyników zawodów.	Ś

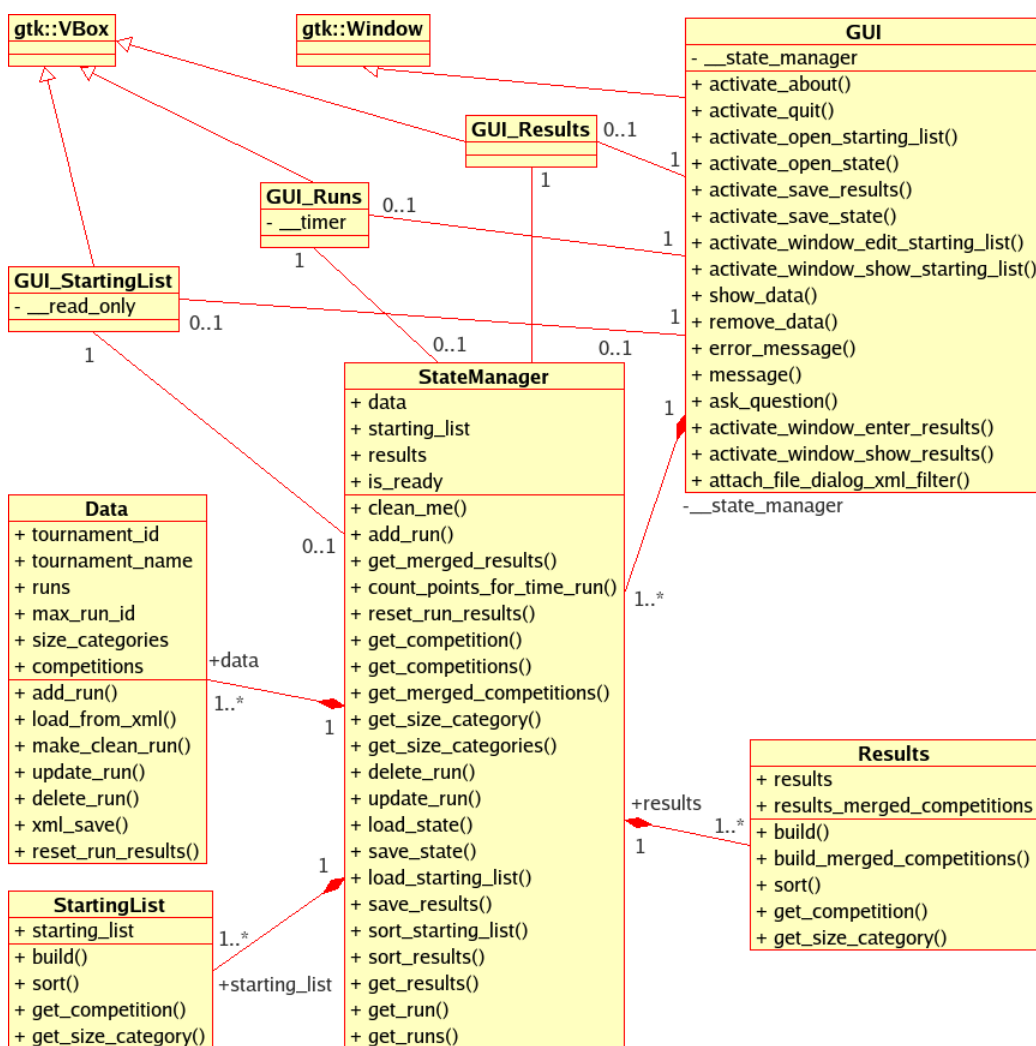
4.5 Ograniczenia środowiskowe

Tabela 4.6: AgileDesk – ograniczenia środowiskowe

Nr	Opis wymagania	Pr.
EC-1	System musi pracować w środowisku Microsoft Windows.	W
EC-2	System powinien pracować w środowisku GNU/Linux.	N
EC-3	System nie powinien wymagać od użytkownika instalacji wielu dodatkowych aplikacji lub bibliotek.	Ś
EC-4	System powinien pracować na architekturze x86.	W
EC-5	System powinien pracować niezależnie od architektury.	N

4.6 Diagram klas

Na diagramie 4.7 wyszczególnione zostały klasy użyte w aplikacji AgileDesk. Klasą łączącą elementy interfejsu użytkownika jest GUI, dziedzicząca po `gtk::Window`. Za strukturę danych oraz stan aplikacji odpowiada klasa `StateManager`. Ona też agreguje w sobie klasy bezpośrednio odpowiedzialne za przechowywane dane, tj. `Data` (dane zawodów, zawodników i biegów, a także obsługa plików XML), `StartingList` (organizacja listy startowej) oraz `Results` (wyniki zawodów).



Rysunek 4.7: AgileDesk – diagram klas

Rozdział 5

Aplikacja AgileDesk

5.1 Instalacja

5.1.1 Wymagania

Aplikacja projektowana i pisana była z postawieniem na niezależność od systemu operacyjnego, na którym ma pracować. Przetestowana pod tym względem była na Microsoft Windows XP oraz GNU/Linux, ale jej możliwość uruchomienia jest ograniczona tak naprawdę tylko przez ograniczenia samego Pythona i biblioteki GTK+.

AgileDesk wymaga Pythona w wersji 2.4 lub 2.5. Najprawdopodobniej bez przeszkód uda się ją uruchomić na planowanej wersji przejściowej 2.6 (nieдоступnej w momencie pisania aplikacji). Źródła jak i przygotowane paczki dla różnych systemów operacyjnych i architektur można pobrać ze strony domowej Pythona [14].

Podstawową wymaganą biblioteką jest GTK+ w wersji 2.10 lub nowszej, która odpowiada za graficzny interfejs użytkownika. Na stronie domowej [15] udostępnione są jej źródła. Paczki dla systemów Unix i GNU/Linux należy zainstalować poprzez menedżer oprogramowania danej dystrybucji. Dla systemów Microsoft Windows przygotowane wersje instalacyjne można pobrać ze strony “*Glade/GTK+ for Windows*” [17].

Ostatnimi wymaganymi bibliotekami są interfejsy GTK+ dla języka Python, czyli PyGTK, PyGobject oraz PyCairo. Tak jak w przypadku GTK+ instalacji na systemach Unix i GNU/Linux należy dokonać przez menedżer oprogramowania. Dla Microsoft Windows przygotowane paczki instalacyjne podane są na stronach PyGTK [19].

5.1.2 Instalacja wymaganych komponentów na GNU/Linux

Instalację wszystkich wymaganych komponentów należy przeprowadzić poprzez menedżer oprogramowania danej dystrybucji. Przykładowa instalacja dla systemu Gentoo Linux:

```
# emerge dev-lang/python
# emerge x11-libs/gtk+
# emerge dev-python/pygtk
# emerge dev-python/pygobject
# emerge dev-python/pycairo
```

5.1.3 Instalacja wymaganych komponentów na Microsoft Windows

Po pobraniu archiwów instalacyjnych ze stron wymienionych w rozdziale 5.1.1 należy po kolei zainstalować je w następującej kolejności:

1. Python (np. `python-2.5.1.msi`);
2. GTK+ (np. `gtk-2.10.11-win32-1.exe`);
3. PyGTK (np. `pygtk-2.10.6-1.win32-py2.5.exe`);
4. PyGobject (np. `pygobject-2.12.3-1.win32-py2.5.exe`);
5. PyCairo (np. `pycairo-1.2.6-1.win32-py2.5.exe`).

Przy domyślnej instalacji (bez zmiany miejsca docelowego) pliki poszczególnych bibliotek zostaną umieszczone w katalogach:

1. Python – `C:\Program Files\Python25`;
2. GTK+ – `C:\Program Files\GTK`;
3. PyGTK, PyGobject i PyCairo w podkatalogach Pythona.

Ostatnim krokiem może być dodanie katalogu z interpreterem `python.exe` (np. `C:\Program Files\Python25`) do ścieżki poszukiwań plików PATH dla danego użytkownika. Nie jest to krok konieczny, ale może ułatwić później pracę z programem.

5.1.4 Konfiguracja AgileDesk

Pliki aplikacji AgileDesk należy bezpośrednio skopiować na dysk twardy do wybranego przez siebie katalogu, przykładowo `C:\agiledesk\` (na tym katalogu dalej się będzie opierać niniejsza instrukcja). Aplikacji nie trzeba konfigurować – domyślne ustawienia powinny być odpowiednie. Jeśli jednak użytkownik chciałby lepiej dostosować zachowanie AgileDesk do swoich potrzeb, to znajdzie konfigurację w pliku `C:\agiledesk\config.py` (plik tekstowy w kodowaniu UTF-8). Wśród istotniejszych zmiennych tam się znajdujących wymienić należy:

DEBUG – boolean (True/False); określa czy mają być drukowane na standardowe wyjście dodatkowe komunikaty o wykonywanych czynnościach (przydatne przy debugowaniu);

WINDOW_WIDTH – integer; początkowa szerokość okna GUI;

WINDOW_HEIGHT – integer; początkowa wysokość okna GUI;

POINTS_FOR_CLICK – integer; ilość punktów przyznawana jednorazowo przy naciśnięciu na przycisk popełnienia błędu przez zawodnika;

5.2 Instrukcja użytkownika

5.2.1 Uruchomienie

Na platformie Microsoft Windows AgileDesk należy przejść do katalogu z aplikacją i uruchomić ją poleceniem:

```
cd C:\agiledesk
C:\Program Files\Python\python.exe agiledesk
```

W przypadku pozostałych systemów operacyjnych (tj. GNU/Linux, Unix) wystarczy uruchomić bezpośrednio plik wykonywalny `agiledesk`.

AgileDesk posiada graficzny interfejs do pełnej komunikacji z użytkownikiem, aczkolwiek korzysta również ze standardowego wyjścia do wyświetlania komunikatów o błędach czy wykonywanych czynnościach (przy włączonej opcji `DEBUG`).

5.2.2 Rozpoczęcie pracy

Pierwszą czynnością po uruchomieniu programu musi być wczytanie:

1. listy startowej (CTRL+I lub z menu *Plik – Importuj listę startową*);
2. stanu aplikacji (CTRL+O lub z menu *Plik – Wczytaj stan aplikacji*);

W obydwu przypadkach zostaną wyświetlone okna wyboru pliku. Przez stan aplikacji rozumie się wcześniej utworzony plik poprzez *Zapisz stan aplikacji* (również z menu *Plik*).

5.2.3 Wyświetlenie i modyfikacja listy startowej

Listę startową, po wybraniu z menu *Zawody – Lista startowa*, można modyfikować przez cały czas uruchomienia programu (rys. 5.1). Należy pamiętać o następujących regułach:

The screenshot shows the AgileDesk application window with the following structure:

- Agility (dług.: 0.0m)**
 - Small (SCT: 20.0s, MCT: 40.0s)**

[kolejność]	[nr startowy]	[przewodnik]	[pies]	[usuń]
6	6	Jan Kowalski	Maks	usuń
 - Medium (SCT: 15.0s, MCT: 30.0s)**

[kolejność]	[nr startowy]	[przewodnik]	[pies]	[usuń]
2	3	Jan Kowalski	Reks	usuń
3	4	Adminowicz Pierwszy	Burek	usuń
- Jumping (dług.: 0.0m)**
 - Medium (SCT: 15.0s, MCT: 30.0s)**

[kolejność]	[nr startowy]	[przewodnik]	[pies]	[usuń]
1	3	Jan Kowalski	Reks	usuń
2	1	Adminowicz Pierwszy	Azor	usuń
- Youngsters (dług.: 0.0m)**
- Zerówka (dług.: 0.0m)**
- Dodaj nowy bieg**

[konkurencja]	[kat. wzrost.]	[kolejność]	[nr startowy]	[przewodnik]	[pies]	[dodaj]
						dodaj

Rysunek 5.1: AgileDesk – ekran modyfikacji listy startowej

- lista jest sortowana przy wczytaniu, więc zmiana pola *kolejność* nie spowoduje zmiany kolejności wierszy na samym ekranie dopóki nie zostanie on ponownie wyświetlony;

- rekordy dodawane są początkowo na koniec listy – zostaną umieszczone w prawidłowym miejscu przy ponownym wczytaniu ekranu (tj. wykonaniu sortowania);
- przy modyfikacji nazwiska przewodnika lub nazwy psa w jednym rekordzie, to w gestii użytkownika leży zmiana tych danych w pozostałych startach tego zawodnika;

5.2.4 Wprowadzanie wyników biegów

Ekran dostępny z menu *Zawody – Wprowadź wyniki* (rys. 5.2) wyświetla aktualną listę startową, ale z polami odpowiadającymi poszczególnym błędom, czasowi i dyskwalifikacji w celu wprowadzania wyników poszczególnych biegów.

Agility (dług.: 0.0m)													
Small (SCT: 20.0s, MCT: 40.0s)													
[kolejność]	[nr startowy]	[przewodnik:]	[pies]	[błędy]	[odmowy]	[za czas]	[czas]	[dis.]					
1	6	Jan Kowalski	Maks	0	+	0	+	0.0	licz	0.0	start	dis.	reset
Medium (SCT: 15.0s, MCT: 30.0s)													
[kolejność]	[nr startowy]	[przewodnik:]	[pies]	[błędy]	[odmowy]	[za czas]	[czas]	[dis.]					
1	3	Jan Kowalski	Reks	0	+	0	+	0.0	licz	0.0	start	dis.	reset
2	4	Adminowicz Pierwszy	Burek	0	+	0	+	0.0	licz	0.0	start	dis.	reset
Jumping (dług.: 0.0m)													
Medium (SCT: 15.0s, MCT: 30.0s)													
[kolejność]	[nr startowy]	[przewodnik:]	[pies]	[błędy]	[odmowy]	[za czas]	[czas]	[dis.]					
1	3	Jan Kowalski	Reks	0	+	0	+	0.0	licz	0.0	start	dis.	reset
2	1	Adminowicz Pierwszy	Azor	0	+	0	+	0.0	licz	0.0	start	dis.	reset

Rysunek 5.2: AgileDesk – ekran wprowadzania wyników biegów

Użytkownik może ręcznie wpisać wartość błędu lub nacisnąć przycisk *+* inkrementując pole o standardową wartość (ustaloną w pliku konfiguracyjnym `config.py` w zmiennej `POINTS_FOR_CLICK`, domyślnie 5).

System umożliwi pomiar czasu dla każdego z biegów niezależnie poprzez przyciski *Start/Stop* przy każdym biegu w kolumnie *[czas]*. Jeśli pole *[czas]* jest wypełnione, system może automatycznie obliczyć ilość punktów karnych związanych z przekroczeniem średniego czasu przebiegu (SCT; widoczne przy nazwie kategorii wzrostowej) lub ewentualnie zdyskwalifikować zawodnika po przekroczeniu maksymalnego czasu przebiegu (MCT). W tym celu użytkownik powinien nacisnąć przycisk *licz*.

W przypadku popełnienia pomyłki i chęci wyzerowania wyników danego biegu lub skasowania dyskwalifikacji, należy nacisnąć *reset*.

5.2.5 Wyniki zawodów i eksport

The screenshot shows the AgileDesk application window with the following data:

Agility + Jumping								
Medium (SCT: 15.0s, MCT: 30.0s)								
	[nr startowy]	[przewodnik]	[pies]	[czas]	[błędy]	[odmowy]	[za czas]	[łącznie]
1	3	Jan Kowalski	Reks	0.0	0	0	0.0	0.0
Agility (dług.: 0.0m)								
Small (SCT: 20.0s, MCT: 40.0s)								
	[nr startowy]	[przewodnik]	[pies]	[czas]	[błędy]	[odmowy]	[za czas]	[łącznie]
1	6	Jan Kowalski	Maks	0.0	0	0	0.0	0.0
Medium (SCT: 15.0s, MCT: 30.0s)								
	[nr startowy]	[przewodnik]	[pies]	[czas]	[błędy]	[odmowy]	[za czas]	[łącznie]
1	3	Jan Kowalski	Reks	0.0	0	0	0.0	0.0
2	4	Adminowicz Pierwszy	Burek	0.0	0	0	0.0	0.0
Jumping (dług.: 0.0m)								
Medium (SCT: 15.0s, MCT: 30.0s)								
	[nr startowy]	[przewodnik]	[pies]	[czas]	[błędy]	[odmowy]	[za czas]	[łącznie]

Rysunek 5.3: AgileDesk – ekran wyników zawodów

W dowolnym momencie można wyświetlić aktualne wyniki zawodów (menu *Zawody – Wyniki zawodów*), przy czym nie zawierają one będą biegów z czasem równym 0,0 (rys. 5.3).

W celu przeniesienia wyników do aplikacji AgileWeb należy dokonać ich eksportu do pliku XML. Po wyborze z menu *Plik – Eksportuj wyniki zawodów* (lub skrót klawiszowy CTRL+E) ukaże się okno wyboru pliku do zapisu.

Rozdział 6

Specyfikacje plików XML do przechowywania i wymiany danych

6.1 Lista startowa

6.1.1 Użycie

Plik XML w tym formacie będzie generowany przez aplikację AgileWeb oraz importowany przez AgileDesk. Zawierać musi dane zawodów (w tym listę konkurencji i kategorii wzrostowych) oraz wszystkie zgłoszenia zawodników i kolejność ich startu.

6.1.2 XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Starting list schema for AgileWeb and AgileDesk.
      Copyright (c) 2007 Krzysztof Kozlowski.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="starting_list" type="StartingListType" />
  <xsd:complexType name="StartingListType">
    <xsd:sequence>
      <xsd:element name="tournament_id" type="xsd:integer" />
      <xsd:element name="tournament_name" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

```

    <xsd:element name="competitions"
      type="CompetitionsType" />
    <xsd:element name="size_categories"
      type="SizeCategoriesType" />
    <xsd:element name="runs" type="RunsType" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CompetitionsType">
  <xsd:sequence>
    <xsd:element name="competition" minOccurs="0"
      maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="id" type="xsd:integer" />
          <xsd:element name="merge_with_competition_id"
            type="xsd:integer" />
          <xsd:element name="order" type="xsd:integer" />
          <xsd:element name="name" type="xsd:string" />
          <xsd:element name="sct_time" type="xsd:decimal" />
          <xsd:element name="mct_time" type="xsd:decimal" />
          <xsd:element name="length" type="xsd:decimal" />
          <xsd:element name="points_type"
            type="xsd:integer" />
          <xsd:element name="has_time" type="xsd:boolean" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SizeCategoriesType">
  <xsd:sequence>
    <xsd:element name="size_category" minOccurs="0"
      maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="id" type="xsd:integer" />
          <xsd:element name="order" type="xsd:integer" />
          <xsd:element name="name" type="xsd:string" />
          <xsd:element name="short_name" type="xsd:string" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

```

```

        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="RunsType">
    <xsd:sequence>
        <xsd:element name="run" minOccurs="0"
            maxOccurs="unbounded">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="user_id" type="xsd:integer" />
                    <xsd:element name="user_full_name"
                        type="xsd:string" />
                    <xsd:element name="user_is_junior"
                        type="xsd:boolean" />
                    <xsd:element name="user_is_debutant"
                        type="xsd:boolean" />
                    <xsd:element name="dog_id" type="xsd:integer" />
                    <xsd:element name="dog_name" type="xsd:string" />
                    <xsd:element name="dog_is_veteran"
                        type="xsd:boolean" />
                    <xsd:element name="competition_id"
                        type="xsd:integer" />
                    <xsd:element name="size_category_id"
                        type="xsd:integer" />
                    <xsd:element name="starting_position"
                        type="xsd:integer" />
                    <xsd:element name="starting_number"
                        type="xsd:integer" />
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

6.1.3 Przykład wygenerowanego pliku

```

<?xml version="1.0" encoding="utf-8"?>
<starting_list
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="specyfikacja_xml_lista_startowa.xsd">

```

```

<tournament_id>1</tournament_id>
<tournament_name>Przykładowe zawody</tournament_name>
<competitions>
  <competition>
    <id>1</id>
    <merge_with_competition_id>0</merge_with_competition_id>
    <order>1</order>
    <name>Agility</name>
    <sct_time>12.5</sct_time>
    <mct_time>25</mct_time>
    <length>12.5</length>
    <points_type>0</points_type>
    <has_time>true</has_time>
  </competition>
</competitions>
<size_categories>
  <size_category>
    <id>1</ID>
    <order>1</order>
    <name>Small</name>
    <short_name>S</short_name>
  </size_category>
</size_categories>
<runs>
  <run>
    <user_id>2</user_id>
    <user_full_name>Jan Kowalski</user_full_name>
    <user_is_junior>false</user_is_junior>
    <user_is_debutant>false</user_is_debutant>
    <dog_id>3</dog_id>
    <dog_name>Azor</dog_name>
    <dog_is_veteran>false</dog_is_veteran>
    <competition_id>1</competition_id>
    <size_category_id>1</size_category_id>
    <starting_position>5</starting_position>
    <starting_number>6</starting_number>
  </run>
  <run>
    <user_id>3</user_id>
    <user_full_name>Onufry Zagłoba</user_full_name>
    <user_is_junior>false</user_is_junior>

```

```

        <user_is_debutant>false</user_is_debutant>
        <dog_id>4</dog_id>
        <dog_name>Burek</dog_name>
        <dog_is_veteran>false</dog_is_veteran>
        <competition_id>1</competition_id>
        <size_category_id>1</size_category_id>
        <starting_position>6</starting_position>
        <starting_number>7</starting_number>
    </run>
</runs>
</starting_list>

```

6.2 Wyniki zawodów

6.2.1 Użycie

Plik XML w tym formacie będzie generowany przez aplikację AgileDesk oraz importowany przez AgileWeb. Zawierać będzie wyniki wszystkich biegów zawodników.

6.2.2 XML Schema

```

<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Tournament results schema for AgileWeb and AgileDesk.
      Copyright (c) 2007 Krzysztof Kozłowski.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="tournament_results"
    type="TournamentResultsType" />
  <xsd:complexType name="TournamentResultsType">
    <xsd:sequence>
      <xsd:element name="tournament_id" type="xsd:integer" />
      <xsd:element name="runs" type="RunsType" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="RunsType">
    <xsd:sequence>

```

```

<xsd:element name="run" minOccurs="0"
  maxOccurs="unbounded">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="user_id" type="xsd:integer" />
      <xsd:element name="dog_id" type="xsd:integer" />
      <xsd:element name="competition_id"
        type="xsd:integer" />
      <xsd:element name="size_category_id"
        type="xsd:integer" />
      <xsd:element name="time" type="xsd:decimal" />
      <xsd:element name="points_for_errors"
        type="xsd:integer" />
      <xsd:element name="points_for_time"
        type="xsd:decimal" />
      <xsd:element name="points_for_denials"
        type="xsd:integer" />
      <xsd:element name="disqualification"
        type="xsd:boolean" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

6.2.3 Przykład wygenerowanego pliku

```

<?xml version="1.0" encoding="utf-8"?>
<tournament_results
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="specyfikacja_xml_wyniki_zawodow.xsd">
  <tournament_id>1</tournament_id>
  <runs>
    <run>
      <user_id>2</user_id>
      <dog_id>3</dog_id>
      <competition_id>4</competition_id>
      <size_category_id>5</size_category_id>
      <time>6.6</time>
      <points_for_errors>0</points_for_errors>
    </run>
  </runs>
</tournament_results>

```

```

    <points_for_time>0.0</points_for_time>
    <points_for_denials>0</points_for_denials>
    <disqualification>>false</disqualification>
</run>
<run>
  <user_id>3</user_id>
  <dog_id>4</dog_id>
  <competition_id>4</competition_id>
  <size_category_id>5</size_category_id>
  <time>6.6</time>
  <points_for_errors>0</points_for_errors>
  <points_for_time>0.0</points_for_time>
  <points_for_denials>0</points_for_denials>
  <disqualification>>false</disqualification>
</run>
</runs>
</tournament_results>

```

6.3 Stan systemu AgileDesk

6.3.1 Użycie

Plik XML w tym formacie będzie używany przez aplikację AgileDesk do przechowywania stanu systemu pomiędzy różnymi jego uruchomieniami. Taki plik musi przechowywać wszystkie informacje dotyczące zawodów, które podlegają tej aplikacji.

6.3.2 XML Schema

```

<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Schema for AgileDesk application state file.
      Copyright (c) 2007 Krzysztof Kozlowski.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="agiledesk_state"
    type="AgileDeskStateType" />
  <xsd:complexType name="AgileDeskStateType">

```

```

<xsd:sequence>
  <xsd:element name="tournament_id" type="xsd:integer" />
  <xsd:element name="tournament_name" type="xsd:string" />
  <xsd:element name="competitions"
    type="CompetitionsType" />
  <xsd:element name="size_categories"
    type="SizeCategoriesType" />
  <xsd:element name="runs" type="RunsType" />
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CompetitionsType">
  <xsd:sequence>
    <xsd:element name="competition" minOccurs="0"
      maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="id" type="xsd:integer" />
          <xsd:element name="merge_with_competition_id"
            type="xsd:integer" />
          <xsd:element name="order" type="xsd:integer" />
          <xsd:element name="name" type="xsd:string" />
          <xsd:element name="sct_time" type="xsd:decimal" />
          <xsd:element name="mct_time" type="xsd:decimal" />
          <xsd:element name="length" type="xsd:decimal" />
          <xsd:element name="points_type"
            type="xsd:integer" />
          <xsd:element name="has_time" type="xsd:boolean" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SizeCategoriesType">
  <xsd:sequence>
    <xsd:element name="size_category" minOccurs="0"
      maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="id" type="xsd:integer" />
          <xsd:element name="order" type="xsd:integer" />
          <xsd:element name="name" type="xsd:string" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

```

```

        <xsd:element name="short_name" type="xsd:string" />
    </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="RunsType">
    <xsd:sequence>
        <xsd:element name="run" minOccurs="0"
            maxOccurs="unbounded">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="user_id" type="xsd:integer" />
                    <xsd:element name="user_full_name"
                        type="xsd:string" />
                    <xsd:element name="user_is_junior"
                        type="xsd:boolean" />
                    <xsd:element name="user_is_debutant"
                        type="xsd:boolean" />
                    <xsd:element name="dog_id" type="xsd:integer" />
                    <xsd:element name="dog_name" type="xsd:string" />
                    <xsd:element name="dog_is_veteran"
                        type="xsd:boolean" />
                    <xsd:element name="competition_id"
                        type="xsd:integer" />
                    <xsd:element name="size_category_id"
                        type="xsd:integer" />
                    <xsd:element name="starting_position"
                        type="xsd:integer" />
                    <xsd:element name="starting_number"
                        type="xsd:integer" />
                    <xsd:element name="time" type="xsd:decimal" />
                    <xsd:element name="points_for_errors"
                        type="xsd:integer" />
                    <xsd:element name="points_for_time"
                        type="xsd:decimal" />
                    <xsd:element name="points_for_denials"
                        type="xsd:integer" />
                    <xsd:element name="disqualification"
                        type="xsd:boolean" />
                    <xsd:element name="is_deleted"

```

```

        type="xsd:boolean" />
    </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

6.3.3 Przykład wygenerowanego pliku

```

<?xml version="1.0" encoding="utf-8"?>
<agiledesk_state
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="specyfikacja_xml_agiledesk_state.xsd">
  <tournament_id>1</tournament_id>
  <tournament_name>Przykładowe zawody</tournament_name>
  <competitions>
    <competition>
      <id>1</id>
      <merge_with_competition_id>0</merge_with_competition_id>
      <order>1</order>
      <name>Agility</name>
      <sct_time>12.5</sct_time>
      <mct_time>25</mct_time>
      <length>12.5</length>
      <points_type>0</points_type>
      <has_time>true</has_time>
    </competition>
  </competitions>
  <size_categories>
    <size_category>
      <id>1</ID>
      <order>1</order>
      <name>Small</name>
      <short_name>S</short_name>
    </size_category>
  </size_categories>
  <runs>
    <run>
      <user_id>2</user_id>
      <user_full_name>Jan Kowalski</user_full_name>
    </run>
  </runs>
</agiledesk_state>

```

```

<user_is_junior>false</user_is_junior>
<user_is_debutant>false</user_is_debutant>
<dog_id>3</dog_id>
<dog_name>Azor</dog_name>
<dog_is_veteran>false</dog_is_veteran>
<competition_id>1</competition_id>
<size_category_id>1</size_category_id>
<starting_position>5</starting_position>
<starting_number>6</starting_number>
<time>6.6</time>
<points_for_errors>0</points_for_errors>
<points_for_time>0.0</points_for_time>
<points_for_denials>0</points_for_denials>
<disqualification>false</disqualification>
<is_deleted>false</is_deleted>
</run>
<run>
  <user_id>3</user_id>
  <user_full_name>Onufry Zagłoba</user_full_name>
  <user_is_junior>false</user_is_junior>
  <user_is_debutant>false</user_is_debutant>
  <dog_id>4</dog_id>
  <dog_name>Burek</dog_name>
  <dog_is_veteran>false</dog_is_veteran>
  <competition_id>1</competition_id>
  <size_category_id>1</size_category_id>
  <starting_position>6</starting_position>
  <starting_number>7</starting_number>
  <time>6.6</time>
  <points_for_errors>0</points_for_errors>
  <points_for_time>0.0</points_for_time>
  <points_for_denials>0</points_for_denials>
  <disqualification>false</disqualification>
  <is_deleted>false</is_deleted>
</run>
</runs>
</agiledesk_state>

```

Rozdział 7

Podsumowanie

7.1 Ocena realizacji wymagań

Projekt, będący częścią niniejszej pracy, dokładnie określił wymagania funkcjonalne stawiane systemowi oraz ich priorytety (*wysoki*, *średni* i *niski*). Poszczególne priorytety zostały opisane w rozdziale 1.5. Wszystkie wymagania o priorytecie *wysoki* oraz *średni* zostały spełnione poprzez ich implementację w realizowanych aplikacjach.

Wśród wymagań o priorytecie *niskim* została zaimplementowana tylko część, gdyż przedstawiają one między innymi możliwy najbliższy rozwój systemu w celu lepszego zaspokojenia potrzeb użytkowników. Wśród niezaimplementowanych reguł funkcjonalnych o priorytecie niskim są dla aplikacji AgileWeb:

- tworzenie konkurencji ze zdobywaniem punktów i uwzględnienie takich w wynikach zawodów (wymaganie FR—Z—16, tabela 2.3);

oraz dla aplikacji AgileDesk:

- drukowanie listy startowej z poziomu aplikacji (wymaganie FR—L—5, tabela 4.1);
- drukowanie wyników zawodów z poziomu aplikacji (wymaganie FR—W—1, tabela 4.2);
- uwzględnienie konkurencji ze zdobywaniem punktów w wynikach zawodów (wymaganie FR—W—10, tabela 4.2)

7.2 Rozwój systemu

Wykonany w ramach pracy magisterskiej system jest jednym z pierwszych w Polsce (o ile nie pierwszym) w całości poświęconych zawodom psów sportowych agility. Został zaprojektowany mając na uwadze specyfikę tych zawodów i oczekiwania potencjalnych użytkowników. Umieszczony na dołączonym do niniejszej pracy nośniku oprogramowania kod źródłowy obydwu aplikacji udostępniony jest na zasadach licencji Wolnego Oprogramowania – GNU General Public License w wersji 2 [26]. Wszystkie użyte przy projektowaniu i wykonywaniu aplikacji biblioteki, narzędzia czy języki programowania także zaliczyć należy do Wolnego Oprogramowania. Dzięki temu system może być dalej rozwijany przez dowolne osoby w różnym kierunku i ku zaspokojeniu różnych potrzeb. Wolność nie tylko używania, ale i modyfikacji oraz redystrybucji systemu była intencją przyświecającą mi przy jego tworzeniu.

Rozbudowa poszczególnych aplikacji może iść w kierunku realizacji wyżej wymienionych wymagań funkcjonalnych w celu lepszego zaspokojenia potrzeb ewentualnego klienta. Innymi aspektami rozwoju może być przeportowanie kodu AgileDesk do języka Python w wersji 3.0, który ma zostać wydany w stabilnej wersji w sierpniu 2008 roku (wersja 3.0 wprowadza szereg zmian w samej strukturze języka Python).

Ciekawą funkcjonalnością, z punktu widzenia potrzeb użytkownika, może być moduł automatycznego kontaktowania się aplikacji AgileDesk z AgileWeb poprzez sieć internet w celu wymiany danych – bez konieczności korzystania z plików XML. AgileDesk mając dostęp do internetu mogłoby w tle uaktualniać dane znajdujące się w bazie danych AgileWeb. Stanowiłoby to również zabezpieczenie podczas zawodów przed utratą już zebranych danych (związaną np. z awarią notebooka z uruchomioną aplikacją). Dzięki zastosowaniu modelu Model–View–Controller przy budowie aplikacji AgileWeb dodanie takiej funkcjonalności ani nie będzie bardzo pracochłonne ani nie będzie wiązać się z koniecznością przebudowy architektury.

Spis rysunków

2.1	Zgłoszenie w zawodach – schemat	8
2.2	Przebieg zawodów – schemat	9
2.3	AgileWeb – Aktorzy w systemie	10
2.4	AgileWeb – przypadek użycia – Logowanie	11
2.5	AgileWeb – przypadek użycia – Zarządzanie kontem	15
2.6	AgileWeb – przypadek użycia – Wyświetlenie zawodów	18
2.7	AgileWeb – przypadek użycia – Zgłoszenie do zawodów	20
2.8	AgileWeb – przypadek użycia – Zarządzanie zawodami	23
2.9	AgileWeb – przypadek użycia – Modyfikacja zawodów	25
2.10	AgileWeb – przypadek użycia – Zarządzanie użytkownikami	29
2.11	AgileWeb – przypadek użycia – Wyświetlenie listy startowej	32
2.12	AgileWeb – przypadek użycia – Zarządzanie zgłoszeniami	33
2.13	AgileWeb – diagram komponentów i podział na warstwy	40
2.14	AgileWeb – diagram związków encji	41
3.1	AgileWeb – ekran rejestracji w systemie	45
3.2	AgileWeb – ekran konta użytkownika	46
3.3	AgileWeb – ekran modyfikacji zgłoszenia w zawodach	46
3.4	AgileWeb – ekran modyfikacji konkurencji	47
3.5	AgileWeb – ekran łączenia kategorii wzrostowych i konkurencji	48
3.6	AgileWeb – ekran listy startowej	49
3.7	AgileWeb – ekran wyników zawodów	50
4.1	Organizacja zawodów – schemat	52
4.2	Biegi w zawodach – schemat	53
4.3	AgileDesk – Aktorzy w systemie	54
4.4	AgileDesk – przypadek użycia – Lista startowa	54
4.5	AgileDesk – przypadek użycia – Wyniki biegów	56
4.6	AgileDesk – przypadek użycia – Obsługa stanu	58
4.7	AgileDesk – diagram klas	64
5.1	AgileDesk – ekran modyfikacji listy startowej	68

5.2	AgileDesk – ekran wprowadzania wyników biegów	69
5.3	AgileDesk – ekran wyników zawodów	70

Spis tabel

2.1	AgileWeb – wymagania funkcjonalne – konto i użytkownicy . . .	35
2.2	AgileWeb – wymagania funkcjonalne – uczestnicy i zgłoszenia do zawodów	36
2.3	AgileWeb – wymagania funkcjonalne – uczestnicy i zgłoszenia do zawodów c.d.	37
2.4	AgileWeb – wymagania funkcjonalne – listy startowe, wyniki końcowe i obsługa danych	38
2.5	AgileWeb – wymagania pozafunkcjonalne	38
2.6	AgileWeb – ograniczenia środowiskowe	39
4.1	AgileDesk – wymagania funkcjonalne – lista startowa	60
4.2	AgileDesk – wymagania funkcjonalne – wyniki biegów i zawodów	61
4.3	AgileDesk – wymagania funkcjonalne – wyniki biegów i zawodów c.d.	62
4.4	AgileDesk – wymagania funkcjonalne – obsługa danych	62
4.5	AgileDesk – wymagania pozafunkcjonalne	63
4.6	AgileDesk – ograniczenia środowiskowe	63

Bibliografia

- [1] PHP: Hypertext Preprocessor. <http://www.php.net>
- [2] PHP License Information. <http://www.php.net/license>
- [3] The Free Software Foundation. <http://www.fsf.org>
- [4] The Free Software Foundation. GPL-Incompatible Free Software Licenses – PHP License, Version 3.01.
<http://www.fsf.org/licensing/licenses>
- [5] CodeIgniter – Open source PHP web application framework.
<http://codeigniter.com>
- [6] CodeIgniter License Agreement.
http://codeigniter.com/user_guide/license.html
- [7] CodeIgniter Installation Instructions.
http://codeigniter.com/user_guide/installation/index.html
- [8] Jesse James Garrett. Ajax: A New Approach to Web Applications.
<http://www.adaptivepath.com/ideas/essays/archives/000385.php>
- [9] xajax PHP Class Library. <http://www.xajaxproject.org>
- [10] xajax – License. http://www.xajaxproject.org/bsd_license.txt
- [11] Python Programming Language. <http://www.python.org>
- [12] Python 2.4.2 license.
<http://www.python.org/download/releases/2.4.2/license/>
- [13] The Free Software Foundation. GPL-Compatible Free Software Licenses – License of Python 2.0.1, 2.1.1, and newer versions.
<http://www.fsf.org/licensing/licenses>
- [14] Download Python Software. <http://www.python.org/download>

- [15] GTK+ – The GIMP Toolkit. <http://www.gtk.org>
- [16] GTK+ FAQ. What is GTK+? [GTK 2.x].
<http://www.gtk.org/faq/#AEN81>
- [17] Glade/GTK+ for Windows.
<http://gladewin32.sourceforge.net/modules/wfdownloads>
- [18] PyGTK: GTK+ for Python. <http://www.pygtk.org>
- [19] PyGTK Downloads. <http://www.pygtk.org/downloads.html>
- [20] MySQL Community Server. <http://www.mysql.com>
- [21] The Apache HTTP Server Project. <http://httpd.apache.org>
- [22] PostgreSQL object-relational DBMS. <http://www.postgresql.org>
- [23] W3C Recommendation. XHTML 1.0 The Extensible HyperText Markup Language (Second Edition). <http://www.w3.org/TR/xhtml1>
- [24] W3C Recommendation. Cascading Style Sheets, level 1.
<http://www.w3.org/TR/REC-CSS1>
- [25] Drew McLellan. Very Dynamic Web Interfaces.
<http://www.xml.com/pub/a/2005/02/09/xml-http-request.html>
- [26] GNU General Public License, version 2
<http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>