

Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Jakub Tlałka

Nr albumu: 292665

Implementacja systemu AI-Arena

Praca licencjacka
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
dra. Janusza Jabłonowskiego
Wydział Matematyki Informatyki i Mechaniki

Maj 2012

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data

Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data

Podpis autora (autorów) pracy

Streszczenie

W pracy przedstawiono implementację systemu AI-Arena służącego do przeprowadzania rozgrywek i turniejów pomiędzy programami komputerowymi. Programy rywalizują ze sobą w wybranych grach, przeliczając i wykonując zagrania zgodnie z algorytmami sztucznej inteligencji. System ma, w zamierzeniu twórców, służyć osobom zainteresowanym sztuczną inteligencją do sprawdzenia swoich umiejętności lub jako pomoc przy badaniach nad tą dziedziną nauki.

Słowa kluczowe

programy walczące, arena, sztuczna inteligencja

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.3 Informatyka

Klasyfikacja tematyczna

D. Software
D.0. General

Tytuł pracy w języku angielskim

Implementation of AI-Arena system

Spis treści

1. Wprowadzenie	5
2. Metodologia Pracy	7
3. Wymagania projektowe	9
3.1. Rozpoznanie problemu	9
3.2. Grupa docelowa	9
4. Podstawowe pojęcia	11
4.1. Gra	11
4.2. Program Walczący - Bot	11
4.3. Sędzia	11
4.4. Reguły	11
4.5. Mecz/Rozgrywka	12
4.6. Nadzorca	12
4.7. Gearman	12
4.8. Konkurs	12
5. Zastosowania w nauce i biznesie	13
6. Podobne platformy	15
6.1. Top Coder	15
6.2. AI Challenge	15
6.3. SIO	16
6.4. SPOJ	16
6.5. MAIN	16
6.6. Porównanie serwisów	17
7. Architektura systemu	19
7.1. Schemat Architektury	19
7.2. Nadzorca	19
7.3. Scheduler	21
7.4. Serwis webowy	22
7.5. Baza danych	23
7.6. Testowanie	24
8. Decyzje technologiczne	27
8.1. Django	27
8.2. Nadzorca	28

8.3. Bepieczeństwo Systemu	29
9. Dokumentacja użytkowa	31
9.1. Przypadki użycia systemu	31
9.1.1. Z udziałem użytkownika niezalogowanego (Gościa)	31
9.1.2. Z udziałem użytkownika zalogowanego	32
9.1.3. Z udziałem moderatora konkursu lub gry	36
10. Harmonogram prac	39
11. Przyszłość projektu	43
12. Podział prac	45
12.1. Jakub Tlałka	45
12.2. Michał Kawiak	46
12.3. Szymon Majewski	47
12.4. Karol Żebrowski	47
13. Podsumowanie	49
14. Spis płyty	51
A. Przykładowe gry	53
A.1. Szachy	53
A.2. Poker	53
A.3. Gra giełdowa	53
A.4. Gra w wybory	53
B. Przykładowe programy	55
C. Przebieg przykładowego turnieju	57
Bibliografia	59

Rozdział 1

Wprowadzenie

Serwis AI-Arena jest uniwersalną platformą przeznaczoną do organizowania konkursów algorytmicznych, których główną domeną jest sztuczna inteligencja, w szczególności jej dział, poświęcony inteligentnym programom walczącym. AI-Arena pozwala również na przeprowadzanie zawodów w stylu ACM, co sprawia, że oferuje dużą elastyczność wyboru rodzaju konkursu.

Od kilkudziesięciu lat odbywają się zawody, mające na celu wyłonienie programów najlepiej grających w popularne gry takie jak szachy czy warcaby. AI-Arena pozwala w łatwy sposób organizować takie konkursy. Szczególnie interesujące jest jego zastosowanie w nauce i w biznesie, ze względu na rosnące zapotrzebowanie na inteligentne programy rozwiązujące problemy w warunkach rywalizacji, jak np. giełda. Twórcy liczą na to, że wraz ze wzrostem jego popularności, AI-Arena skupi społeczność osób zafascynowanych dziedziną sztucznej inteligencji i przyczyni się do rozwoju tej gałęzi nauki.

Projekt jest udostępniony w serwisie Github na prawach otwartej licencji GPL. Każda osoba chętna do pomocy przy jego rozwoju może uzyskać prawa zapisu i samodzielnie usprawniać jego działanie.

Rozdział 2

Metodologia Pracy

Aby usprawnić proces tworzenia systemu AI-Arena, autorzy postanowili na początku prac nad systemem wybrać odpowiednią do potrzeb metodologię pracy. Początkowo rozważane były następujące metodologie: Agile Uniformed Process, Extreme Programming, Scrum. Przy dokonywaniu wyboru kierowano się następującymi kryteriami:

- Ze względu na przebieg prac w małym zespole (czterooosobowym) wybrana metodyka powinna być metodyką lekką.
- Jako podstawową jednostkę pracy przyjęto tygodniowe iteracje z uwagi na cotygodniowe spotkania na proseminarium.
- Ważne było aby każdy z członków mógł samodzielnie podjąć decyzję o ilości pracy jaką wykona w najbliższym tygodniu, gdyż w trakcie tworzenia projektu wszyscy członkowie kontynuowali studia i musieli wypełniać związane z tym obowiązki,
- Istotna była możliwość pracy w dogodnych dla siebie godzinach, niezależnie od pozostałych członków zespołu, zatem praca nad projektem miała przebiegać w sposób rozproszony.
- Wybrana metodyka powinna ponadto pozwalać na łatwe śledzenie zmian w projekcie, oraz ułatwiać zapewnianie kontroli jakości.

Żadna z początkowo rozważanych metodologii nie spełniała wszystkich powyższych wymagań. Zdecydowano się na własne rozwiązanie, inspirowane metodologią Scrum. Jest metodologią typu lekkiego, która okres pracy na krótsze podokresy zwane sprintami. Podczas każdego takiego sprintu, trwającego około kilku tygodni, realizowana jest pewna część funkcjonalności.

Ponieważ zajęcia z przedmiotu ZPP odbywały się cotygodniowo, zdecydowano się przyjąć tydzień między zajęciami jako odpowiednik sprintu. Podczas zajęć, ustalany był zakres pracy przydzielony poszczególnym osobom na przyszły tydzień, oraz weryfikowana była praca wykonana w tygodniu poprzednim. Prowadzący zgłaszał uwagi do niektórych funkcjonalności, dyskutowane były również dokumenty powstające przy okazji pracy nad projektem, przede wszystkim niniejsza praca licencjacka. Zapis przebiegu zajęć wraz z celami na przyszły tydzień, nazywany minutkami, był przechowywany w systemie Google Documents, tak by każdy z członków zespołu oraz prowadzący zajęcia posiadał do niego dostęp.

Dodatkowo organizowane były spotkania samego zespołu. Odbywały się one również raz w tygodniu i miały na celu kontrolowanie przebiegu prac i zadbanie o to by zostały wykonane zaplanowane zadania. Spotkania odbywały się w budynku Wydziału Matematyki Informatyki i Mechaniki UW, bądź jako konferencje video z wykorzystaniem technologii Google Hangouts lub Skype. Ich przebieg nadzorował Lider Projektu, który był odpowiedzialny również za realizowanie założeń przyjętej metodologii pracy.

Podział obowiązków między członków zespołu był wynikiem porozumienia między członkami i był zatwierdzany przez Lidera Projektu, a następnie przedstawiany prowadzącemu zajęcia.

Do bieżącej kontroli stanu projektu użyty został system Redmine umożliwiający tworzenie zadań i przydzielanie ich członkom zespołu. Tworzenie nowych zadań, a następnie ich uaktualnianie należało do obowiązków każdego członka zespołu. W trakcie trwania prac okazało się jednak, iż ilość czasu wymagana do koordynowania projektu za pomocą Redmine'a jest nieproporcjonalnie duża i podjęto decyzję o częściowym zarzuceniu korzystania z tego serwisu.

Kod projektu, oraz dokumenty takie jak praca licencjacka, zostały umieszczone w repozytorium Gita. Git to system kontroli wersji dbający o archiwizację kodu oraz koordynację równoległych zmian wszystkich członków zespołu. Repozytorium jest publiczne i dostępne online w serwisie GitHub. Wszystkie zmiany dokonywane przez członków zespołu są archiwizowane, tak by można je było odtworzyć. Możliwe jest również jednoczesne rozwijanie projektu w kilku niezależnych kierunkach, a następnie połączenie efektów pracy.

Z uwagi na niewielką liczebność zespołu zdecydowano się nie używać narzędzi wspomagających śledzenie zmian. Do obowiązków każdego członka zespołu należało upewnienie się, że po zmianach przez niego wprowadzonych projekt uruchamia się bez wystąpienia nowych problemów (regresji). W przypadku wystąpienia regresji mechanizmy kontroli wersji udostępniane przez system Git pozwoliłyby na odtworzenie starszej wersji projektu.

Częste spotkania projektowe umożliwiały członkom zespołu bieżące śledzenie aktualnych zmian. W czasie takich spotkań, każdy z członków relacjonował postęp prac, który następnie był weryfikowany przez Lidera Projektu, bądź też przez innego członka zespołu. Pozwoliło to na bieżąco wyznaczać priorytety poszczególnych zadań oraz kierunek rozwoju serwisu.

Rozdział 3

Wymagania projektowe

3.1. Rozpoznanie problemu

Sztuczna inteligencja jest bardzo popularną dziedziną informatyki. Znajduje zastosowania w kolejnych gałęziach tej nauki. Nic dziwnego, że coraz więcej programistów pragnie pogłębiać swoją wiedzę w tym zakresie. Potrzeba ta nie jest jednak w pełni zaspokojona. W Internecie istnieje wiele platform wspierających naukę algorytmiki (przede wszystkim popularne platformy przygotowujące do konkursów ACM), jednak nie ma żadnego odpowiednika platformy umożliwiającej naukę sztucznej inteligencji.

W odpowiedzi na tę potrzebę, autorzy zdecydowali się stworzyć serwis AI-Arena umożliwiający porównanie programów walczących. Program walczący można opisać jako program podejmujący decyzje w zmieniających się warunkach, rywalizujący z innymi programami według pewnych ściśle określonych zasad. Tworzenie efektywnych programów walczących jest częścią pracy nad rozwojem sztucznej inteligencji. Serwis AI-Arena ułatwia nie tylko rywalizację programistów, ale także wymianę ich doświadczeń z dotychczasowej pracy nad programami walczącymi.

3.2. Grupa docelowa

Jak wynika z dotychczasowego określenia problemu, grupę docelową stanowią programiści, którzy są zainteresowani tematem sztucznej inteligencji. Można się zatem spodziewać, że część tej grupy, zafascynowana również bezpieczeństwem systemów komputerowych, będzie chciała sprawdzić zabezpieczenia systemu. Należało więc ze zdwojonym wysiłkiem zapewnić szeroko rozumiane bezpieczeństwo.

Wygląd serwisu był również zdeterminowany przez charakter grupy docelowej. Autorzy zdecydowali, że powinien być przede wszystkim schludny i czytelny. Nie było potrzeby umieszczania efektownych animacji, bądź wystylizowanych obrazków, które spowolniłyby działanie strony i powodowały dezorientację użytkownika.

Rozdział 4

Podstawowe pojęcia

4.1. Gra

Gra oznacza pojedynczy obszar rywalizacji programów walczących na ściśle określonych zasadach. Zasady są opisane w dokumencie tekstowym zwanym Regulami, który powinien być publicznie dostępny. Na Grę składa się również program Sędziego, który jest implementacją tych Regul. Gra może być jedno, bądź wieloosobowa. W przypadku jednoosobowej gry mamy do czynienia z rozwiązywaniem zagadnień algorytmicznych bez wprowadzania elementu rywalizacji z przeciwnikiem (problem algorytmiczny typu on-line). W warunkach serwisu AI-Arena graczami są programy komputerowe, nazywane Botami bądź Programami Walczącymi. Nie ma ograniczeń dotyczących tematyki Gry. Jedynym warunkiem jest to, by jej zasady dały się zapisać w postaci programu weryfikującego, zwanego Sędzią.

4.2. Program Walczący - Bot

Program napisany w jednym z obsługiwanych przez serwis języków programowania, musi być przypisany do konkretnej Gry dostępnej w serwisie. Uczestniczy w rozgrywkach (Meczach) z innymi botami przypisanymi do tej Gry. Ma za zadanie przetwarzać informacje o dotychczasowym przebiegu rozgrywki i produkować kolejne posunięcia, zgodne z regułami Gry.

4.3. Sędzia

Jest to program związany z daną Grą. Ma on za zadanie kontrolować przebieg Meczu, weryfikować zagrywki graczy, informować ich o aktualnym stanie, oraz ustalać wynik Meczu, zgodnie z zasadami związanymi z Grą, opisanymi w Regulach. Kod Sędziego jest publiczny i dostępny wszystkim osobom korzystającym z serwisu.

4.4. Reguły

Dokument udostępniany publicznie, opisujący zasady gry, oraz protokół formatu komunikatów przesyłanych między Botami a Sędzią. Powinien być zgodny z implementacją programu Sędziego, oraz zapewniać możliwość przeprowadzenia dowolnej rozgrywki w obrębie zasad Gry.

4.5. Mecz/Rozgrywka

Jest rozgrywany między Botami, w obrębie konkursu lub poza nim. Jego przebieg jest kontrolowany przez Sędziego. Za przesyłanie komunikatów między Botami a Sędzią oraz odebranie od Sędziego rezultatu meczu, odpowiedzialny jest Nadzorca.

4.6. Nadzorca

Program będący integralną częścią serwisu AI-Arena. Jego zadaniem jest uruchamianie programów Botów i Sędziego podczas Meczu, a następnie transportowanie między nimi komunikatów tekstowych zgodnych z ustalonym protokołem. Nadzorca odbiera wynik Meczu od Sędziego i przekazuje go do Gearmana (narzędzie używane do kolejkovania zadań) w celu zapisania rezultatu w bazie. Dodatkowo Nadzorca kontroluje czas i pamięć zużywaną przez Boty, tak by nie przekroczyły dozwolonych limitów. Informacje o ich zużyciu wraz z logiem rozgrywki generowanym przez Sędziego, Nadzorca zwraca do Gearmana.

4.7. Gearman

Zewnętrzne narzędzie kolejkovania zadań i transportu danych. W AI-Arena służy jako most między Nadzorcą, a Bazą danych. Kolejkuje Mecze do rozegrania, zleca ich rozegranie Nadzorcy, a następnie zapisuje do bazy przekazany przez Nadzorcę rezultat i dodatkowe informacje o przebiegu rozgrywki.

Strona projektu Gearman:

<http://gearman.org>

4.8. Konkurs

Konkurs może mieć określoną datę zakończenia, bądź być tak zwanym konkursem stałym, w którym nie ma ostatecznego terminu wysyłania rozwiązań. Dla każdego konkursu tworzony jest ranking, bądź drabinka rozgrywek. Określają one kolejność botów, w szczególności zwycięzcę, bądź aktualnego lidera konkursu. W przypadku konkursu stałego istnieje ranking, który jest posortowany po sumie zdobytych przez Bota punktów w meczach, przy czym każdy Bot powinien mieć taką samą liczbę rozegranych meczy.

Rozdział 5

Zastosowania w nauce i biznesie

Zastosowania algorytmów Sztucznej Inteligencji sięgają prawie wszystkich obszarów nie tylko Internetu ale i codziennego życia. Serwis AI-Arena pomaga rozwijać gałąź tej nauki związaną z rywalizacją.

Najprostsze przykłady rywalizacji to oczywiście wszelkiego rodzaju gry i sporty. Obecnie komputery są w stanie wygrywać z człowiekiem w większości gier takich jak szachy, warcaby itp. Warto wspomnieć mecz szachowy mistrza świata Garriego Kasparowa z komputerem Deep Blue czy teleturniej Va Banque z udziałem komputera Watson - w obu przypadkach zwyciężyła sztuczna inteligencja (http://www.benchmark.pl/aktualnosci/Superkomputer_IBM_wystapil_w_teleturnieju-32781.html - dane z 04/2012). Także gry komputerowe korzystają z coraz bardziej wyrafinowanej sztucznej inteligencji. Zaczęto doceniać znaczenie metod naukowych również w sporcie, w celu opracowania optymalnych strategii. Z powodzeniem wykorzystuje się komputer do badania statystyk w amerykańskiej lidze baseballowej (<http://research.sabr.org/journals/online/38-brj-1976/165-computers-in-baseball-analysis> - dane z 04/2012). Prawdopodobnie jest także kwestią czasu analizowanie gry wirtualnych zespołów kierowanych sztuczną inteligencją, a następnie wykorzystywanie obserwacji do poprawy gry prawdziwej drużyny.

Rywalizacja może być wykorzystana również jako metoda rozwiązywania problemów. Przykładem takiego podejścia są algorytmy genetyczne, w których najlepsze jednostki pozostają w obiegu, cały czas udoskonalając swoje podejście do rozwiązywania danego problemu.

Serwis AI-Arena może mieć duże zastosowanie w biznesie. Firmy nieustannie rywalizują między sobą w walce o klienta. Serwis umożliwia symulowanie takiej rywalizacji i dzięki temu odkrywanie skutecznych algorytmów sztucznej inteligencji, które będą podejmowały decyzje decydujące o sukcesie wykorzystującej je firmy. Można także symulować inwestowanie funduszy na przykład na giełdzie. Banki wykorzystują sztuczną inteligencję do organizacji operacji finansowych i zarządzania środkami. Sztuczna inteligencja odnosi już sukcesy w rywalizacji z ludźmi w symulacji obrotów finansowych (<http://news.bbc.co.uk/2/hi/business/1481339.stm> - dane z 04/2012).

Innym przykładem zastosowania AI-Arena są działania wojenne. Serwis może pomóc w szukaniu algorytmów, które będą potrafiły adaptować się do różnych warunków i w zależności od nich sugerować najlepsze strategie i taktyki w walce z przeciwnikiem.

Rozdział 6

Podobne platformy

Obecnie istnieją serwisy internetowe podobne do AI-Arena. Oto kilka z nich:

6.1. Top Coder

(<http://topcoder.com/tc>)

Bardzo popularny serwis organizujący różnego rodzaju konkursy programistyczne. Jednym z nich są tzw Marathon Matche, podczas których uczestnicy wysyłają programy, które starają się najbardziej optymalnie rozwiązać dany problem, przy czym nie istnieje rozwiązanie całkowicie optymalne. W terminach serwisu AI-Arena, taki typ zadania można zapisać jako grę jednoosobową.

Głównym założeniem serwisu TopCoder jest umożliwienie pracodawcom przetestowanie ewentualnych przyszłych pracowników, nawet jeszcze przed rozpoczęciem rekrutacji. Pozwala na szybkie wyeliminowanie sporej liczby kandydatów, którzy okażą się po prostu nieefektywni w przyszłej pracy.

Użytkownicy serwisu mają zaś możliwość zmierzyć się z praktycznymi problemami, które często okazują się być trudne obliczeniowo. Mogą się więc nauczyć stosowania różnego rodzaju heurystyk czy też algorytmów probabilistycznych, co odróżnia platformę TopCoder od innych serwisów nastawionych konkretnie na przygotowanie do konkursów ACM.

6.2. AI Challenge

(<http://aichallenge.org>)

Serwis organizujący w sposób cykliczny zawody dla programów walczących. Najczęściej ok 2 konkursy rocznie. Każdy konkurs ma określony czas trwania i nie można w nim uczestniczyć po jego zakończeniu.

Ciekawą stroną (niektórych) gier publikowanych na AI Challenge jest możliwość wizualnego śledzenia zmian. Pozwala to na łatwiejsze śledzenie zachowania programu walczącego jako całości i ewentualne usunięcie niedoskonałości sztucznej inteligencji. Poza tym generuje również ciekawe animacje obrazujące przebieg partii.

Do tej pory (maj 2012 r.) za pośrednictwem serwisu AI Challenge zostały przeprowadzone 3 konkursy. Najwcześniejszy z nich odbył się na jesieni 2010 r. Ostatni zaś zakończył się w grudniu 2011 r. i od tamtego czasu nie ma oficjalnych informacji na temat przyszłych potyczek.

6.3. SIO

(<http://sio.mimuw.edu.pl>)

Projekt od lat wykorzystywany do organizowania konkursów algorytmicznych, w szczególności polskiej Olimpiady Informatycznej, ale też Międzynarodowej Olimpiady Informatycznej. Jest to framework, który można wykorzystywać również w wersji lokalnej, nie tylko jako serwis internetowy.

Serwis ten umożliwia jednak wysyłanie rozwiązań wyłącznie w trakcie trwania zawodów, które odbywają się ok. 1-2 razy w roku. Sprawdza się jako platforma do organizacji konkursów, jednak nie nadaje się do nauki algorytmiki.

6.4. SPOJ

(<http://spoj.pl>)

Serwis zawierający dużą bazę zadań algorytmicznych dostępnych do rozwiązywania użytkownikom. Zadania nie mają określonego terminu rozwiązywania. Istnieje ranking biorący pod uwagę liczbę wszystkich rozwiązanych przez użytkowników zadań.

Platforma ta umożliwia przygotowanie się do zawodów algorytmicznych jak np. Olimpiada Informatyczna, czy ACM. Za jej pośrednictwem organizowane są również konkursy: zarówno oficjalne (HSPL - High School Programming League), jak i nieoficjalne - towarzyskie. Można ponadto zamówić organizację konkursu za pośrednictwem serwisu SPOJ.

Na co dzień jednak główną zaletą tej witryny jest dostępność dużej liczby problemów, na które można nadsyłać rozwiązania w trybie ciągłym.

6.5. MAIN

(<http://main.edu.pl>)

Posiada dużą bazę zadań z olimpiad informatycznych, ale także kursy umożliwiające pogłębienie wiedzy algorytmicznej. Jest to popularna witryna wśród uczniów szkół licealnych i gimnazjalnych przygotowujących się do konkursów informatycznych. Umożliwia ona wysyłanie rozwiązań do problemów z ubiegłych olimpiad i innych konkursów.

Wadą serwisu jest jednak ograniczenie bazy zadań wyłącznie do problemów archiwalnych z konkursów. Ogranicza to w pewnym stopniu możliwość wszechstronnego przygotowania się do konkursów. Istnieją jednak serwisy (np. wyżej wymieniony SPOJ), które łączą tę lukę.

6.6. Porównanie serwisów

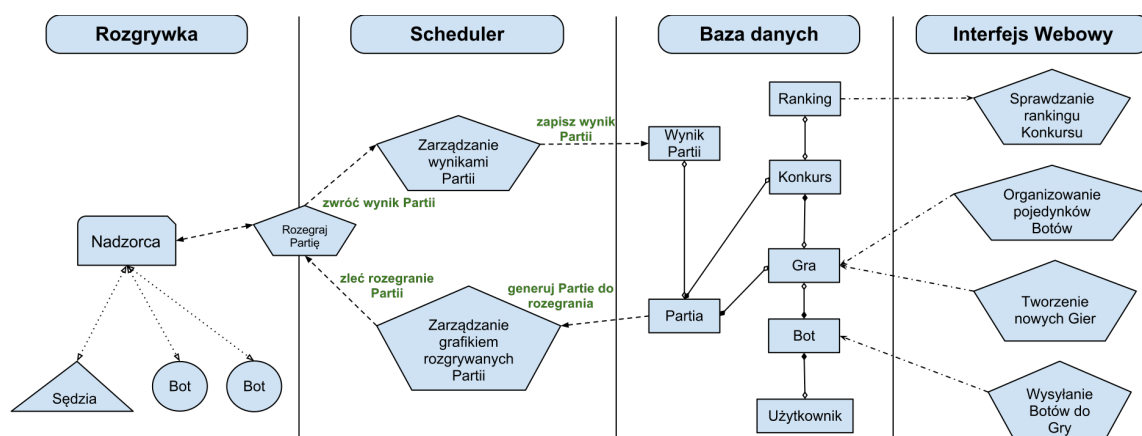
	tematyka	typ konkursów	baza zadań
Top Coder	zadania algorytmiczne, problemy optymalizacyjne, architektura, GUI i inne	okresowe, kilkaset w ciągu roku, często z nagrodami pieniężnymi	bardzo duża baza z przeszłych zawodów i nie tylko
AI Challenge	programy walczące i sztuczna inteligencja	okresowe, ok. 2 razy w roku	konkurs co pół roku
SPOJ	problemy algorytmiczne	stałe i okresowe, ok. raz w roku	duża baza zadań
AI-Arena	programy walczące	stałe i okresowe	baza gier moderowana przez użytkowników
SIO	algorytmy, wyłącznie zadania konkursowe	okresowe, ok. raz w roku	zadania dostępne tylko podczas zawodów
MAIN	algorytmy, stare zadania konkursowe	stałe	archiwalne zadania z najważniejszych zawodów informatycznych

Rozdział 7

Architektura systemu

Na serwis AI-Arena składają się cztery warstwy: Nadzorcy, Scheduler, Bazy Danych i Interfejsu Webowego.

7.1. Schemat Architektury



7.2. Nadzorca

Nadzorca jest jądrem serwisu AI-Arena. Warstwa nadzorcy jest odpowiedzialna za uruchamianie rozgrywek pomiędzy wybranymi graczami, zbieranie informacji o ich wynikach i przekazywanie ich do warstwy schedulera. Nadzorca jest skryptem napisanym w pythonie, którego najważniejszą częścią jest metoda play. Metoda ta przyjmuje jako argumenty uruchamialne pliki Sędziego i programów grających, oraz limity czasowy i pamięciowy dla każdego programu grającego. Następnie metoda play przeprowadza odpowiednią rozgrywkę, zwracając jako wynik słownik zawierający:

- Ciąg liczb oznaczający przydzielone przez sędziego punkty za rozgrywkę
- Informacje na temat przebiegu rozgrywki
- Czas jaki zużyły programy walczące
- Pamięć RAM jaką wykorzystywały programy walczące

Rozgrywka zaczyna się przez uruchomienie programów walczących, oraz Sędziego, kontrolującego przebieg rozgrywki. Komunikacja między sędzią a poszczególnymi programami odbywa się za pośrednictwem nadzorcy, który odczytuje komunikaty ze standardowego wyjścia programów i wypisuje informacje zwrotne na ich standardowe wejście. Format komunikatów od Sędziego i botów powinien być wyspecyfikowany dla każdej gry, przy czym powinien on być zgodny z poniższym protokołem:

- Bot wysyła komunikaty TYLKO do Sędziego (za pośrednictwem Nadzorcy)
- Każdy komunikat od Sędziego musi być potwierdzony komunikatem zwrotnym od Bota.
- Komunikaty wysyłane przez Sędziego mają następujący format: zaczynają się od nawiasów kwadratowych, w których znajduje się lista garczy oddzielona przecinkami. Następnie znajduje się wiadomość, która zostanie przekazana odpowiednim botom. Komunikat musi kończyć się znakiem nowej linii, np: '[1,2,3,4,5]INIT'. Komunikaty muszą być jednolinijkowe i nie mogą zawierać żadnych znaków spoza 7-bitowego kodowania ASCII.
- Sędzia otrzymuje komunikaty zwrotne od botów w takiej kolejności w jakiej zostali wylistowani.
- Komunikaty od nadzorcy do botów i do Sędziego również kończą się znakiem nowej linii. Dzięki temu można przysyłać komunikaty wielowierszowe.
- Jeśli Sędzia chce wysłać komunikat do wszystkich może zacząć ALBO od wylistowania wszystkich graczy, ALBO użyć skrótu notacyjnego: '[0]'.
 - Ponadto, jeśli Sędzia stwierdził, że gra się zakończyła i chce poinformować wszystkich o tym, że nastąpił koniec gry wysyła do Nadzorcy komunikat o treści "[0]END". Następnie powinien wysłać komunikat zawierający punktację dla wszystkich graczy w postaci: '[score1, score2, ...]'.
- Jeśli Sędzia chce zakończyć działanie któregoś z graczy należy wysłać do niego wiadomość KILL (np. "[4]KILL"). Można też w ten sposób zabić większą liczbę graczy lub nawet wszystkich ("[0]KILL")
- Jeśli Sędzia wyśle komunikat do Bota którego działanie zostało przerwane (czy to na skutek wewnętrznego błędu, czy przez zabicie przez Sędziego), dostanie komunikat zwrotny '_DEAD_' (wysłany przez Nadzorcę). Boty powinny unikać wysyłania tego komunikatu, gdyż mogą zostać uznane przez Sędziego za martwe.

Należy dodać, iż serwis nie definiuje maksymalnej dozwolonej liczby programów grających przeciw sobie. Warto jednak wspomnieć, iż przy większej ilości uczestników tworzenie gier dla bardzo wielu uczestników może spowodować problemy wydajnościowe. W praktyce zaleca się więc (szczególnie w turniejach typu "każdy z każdym") ograniczenie liczby graczy do 4,

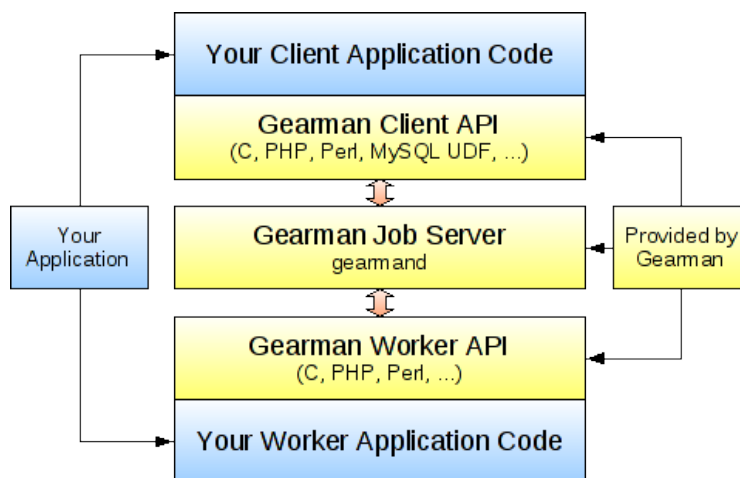
Innym z zadań nadzorcy jest odczytywanie informacji z standardowego strumienia błędów Botów i Sędziego, a następnie przekazanie ich warstwie Schedulera. Pozwala to na umożliwienie użytkownikowi dostęp do informacji które jego program wypisywał na stderr. Funkcjonalność ta jest realizowana przez Nadzorcę, poprzez stworzenie dodatkowego wątku odpowiedzialnego za czytanie informacji pojawiających się na strumieniu błędów Sędziego i Botów. Wątek ten przy użyciu polecenia select czeka na możliwość nieblokującego odczytu

na odpowiednich deskryptorach, a gdy taka możliwość się pojawia odczytuje informacje aż do opróżnienia potoku związanego z danym deskryptorem.

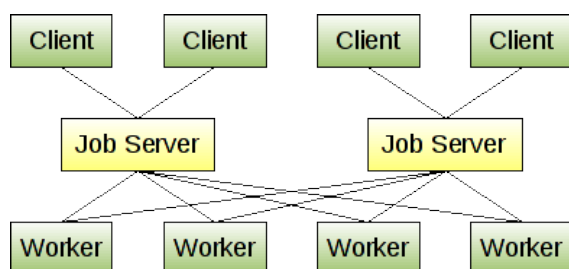
Nietrywialnym problemem jest wybór mechanizmu odpowiedzialnego za ograniczanie i zbieranie informacji o zużywanych przez Sędziego i boty zasobach systemowych. Dostępnych metod jest wiele, jednak nie wszystkie są możliwe do zastosowania w przypadku systemu AI-Arena. Mechanizm ten, jak również motywacje do jego wyboru zostały przedstawione w rozdziale 8.1.

7.3. Scheduler

Do realizacji warstwy Schedulera używany jest program Gearman. Jest on lekkim frameworkiem służącym do przydzielania zadań do wykonania określonym maszynom lub procesom. Posiada prosty interfejs i jest łatwy w użyciu. Każda aplikacja korzystająca z tego frameworku składa się z trzech części: klienta, workera oraz serwera. Serwer jest dostarczany przez aplikację Gearmand (v. 0.26). W naszym przypadku rolę klienta pełni część webowa serwisu zlecająca serwerowi rozegranie odpowiedniej Rozgrywki, podając jako argumenty grę, boty i Sędziego Rozgrywki. Następnie serwer wysyła żądanie do tzw. "workera" zajmującego się wyłącznie wykonywaniem zleceń. Workerów może być wiele i mogą działać na fizycznie różnych maszynach. Problemem równomiernego obciążenia workerów zamuje się serwer. W przypadku AI-Areny funkcjonalność workera ogranicza się do uruchomienia nadzorca z odpowiednimi parametrami, żeby po zakończeniu Rozgrywki zapisać wyniki do bazy danych.



Rysunek 7.1: Aplikacja Gearmana



Rysunek 7.2: Klaster Gearmana

Warstwa Schedulera umożliwia zakolejkowanie kilku Meczy do rozegrania np. w razie dużego ruchu w serwisie lub gdy wymaga tego organizacja Turnieju, oraz wykonanie tych zadań w sposób asynchroniczny. Gearman łączy część webową serwisu z nadzorcą, przez co obie części funkcjonują i mogą być rozwijane niezależnie. Umożliwia także użycie wielu klientów, serwerów i workerów tworzących klaster. Należy dodać, iż w przypadku faktycznego wykorzystania wielu serwerów i workerów całkowita

kontrolę nad nimi sprawuje aplikacja Gearman. Dbą ona między innymi o równomierne wykorzystanie zasobów. Użycie zapasowego serwera pomogłoby ustabilizować pracę serwisu, a uruchomienie kilku workerów na wielordzeniowej maszynie - znacznie ją przyspieszyć.

7.4. Serwis webowy

Serwis AI-Arena jest platformą internetową. Użytkownik ma zatem, po zalogowaniu się, całodobowy dostęp do zasobów zgromadzonych w serwisie. Podstawowe funkcjonalności serwisu można podzielić na następujące kategorie:

1. Tworzenie gry lub konkursu - Użytkownik, który ma pomysł na ciekawą grę ma możliwość dodania jej bez konieczności uzyskania akceptacji administratora strony. Staje się on automatycznie moderatorem dodanej przez siebie gry. Warunkiem dodania gry jest napisanie kodu źródłowego Sędziego oraz szczegółowych zasad gry. Kod źródłowy Sędziego jest dostępny dla wszystkich zalogowanych użytkowników serwisu. Ma to pozwolić uniknąć stronniczych sędziów, bowiem wszelkie nieprawidłowości mogą być zgłaszane do administracji strony, która ma możliwość usunięcia gry.
2. Wysłanie bota do gry lub konkursu - Użytkownik ma możliwość napisania własnego programu grającego w grę zdefiniowaną przez innego użytkownika. Program ten (Bot) musi ściśle spełniać warunki narzucone przez reguły gry. W szczególności musi również spełniać limity czasu i pamięci wyznaczone w specyfikacji gry.

W przypadku, gdy bot przekroczy limit czasu lub pamięci, bądź też naruszy reguły gry w dowolny inny sposób (np. poprzez wysłanie komunikatu sformatowanego w niepoprawny sposób lub wykona niedozwoloną akcję - przykładowo utworzy dodatkowy wątek potomny) będzie natychmiast przerywany i będzie to traktowane jako automatyczna przegrana z minimalną ilością punktów.

Należy zaznaczyć, iż sędzia ma możliwość przerywania wykonania dowolnego bota, jeśli stwierdzi, iż narusza on warunki gry.

Serwis webowy jest zaimplementowany w Django. Ponadto części luźno związane z serwisem, jak np. sędziowie gier mogą być napisane również w C, C++ czy w Javie. W celu zwiększenia komfortu użytkownika serwisu, pewne funkcjonalności zostały również napisane w JavaScriptcie z wykorzystaniem bibliotek JQuery i JQuery-UI. Naturalnie wykorzystane zostały również arkusze stylów CSS. Dodatkowo zostały wykorzystane pewne standardowe programy linuxowe, jak np. Makefile.

Oprócz podstawowych funkcjonalności, ściśle związanych z charakterem serwisu, udostępniane są ponadto następujące operacje:

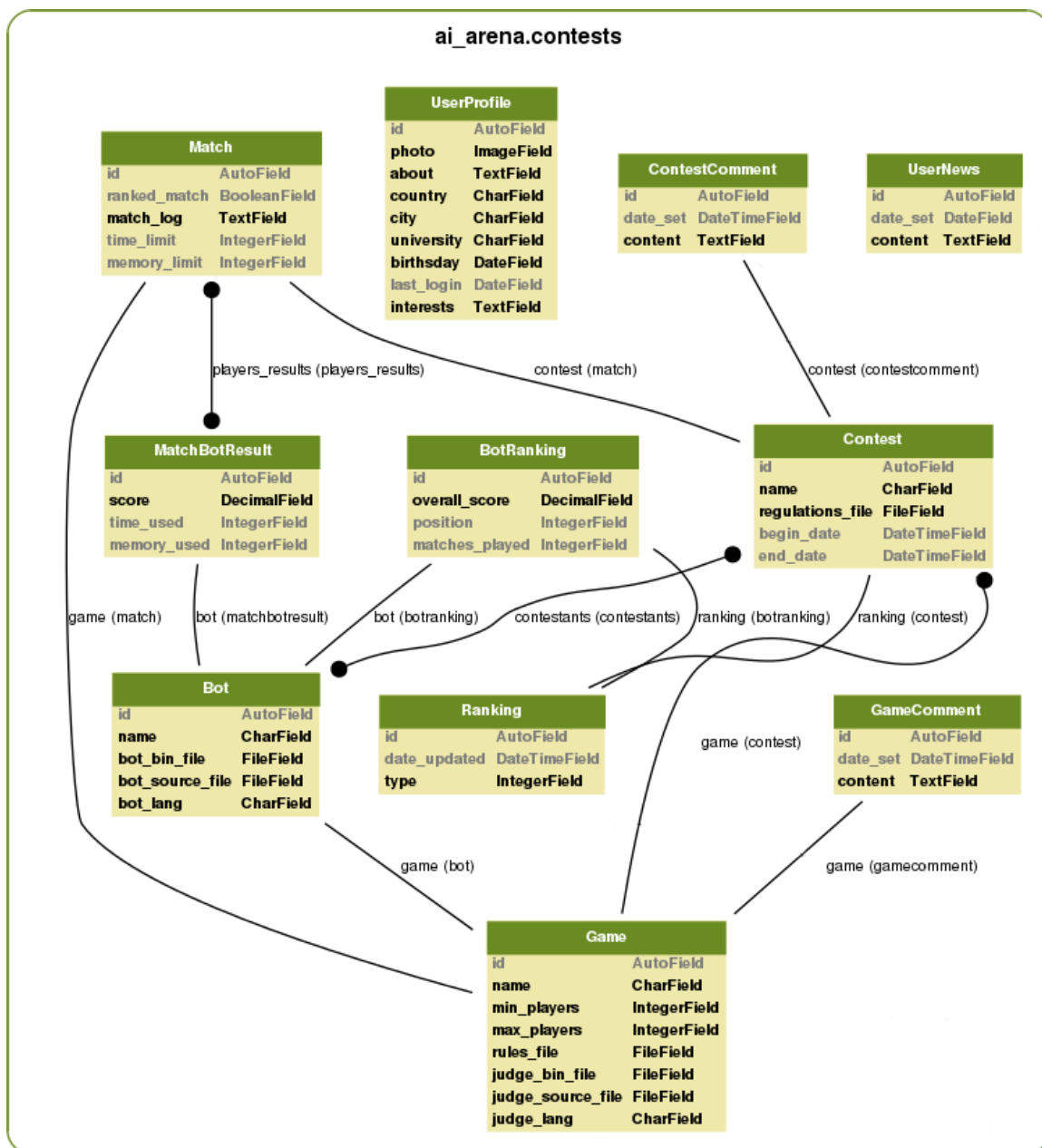
- Wyświetlenie i edycja strony profilowej
- Wyświetlenie kodu źródłowego Sędziego
- Udostępnienie kodu źródłowego wysyłanego bota
- Wyświetlenie udostępnionego kodu źródłowego
- Wyświetlenie rankingu konkursu

- Uruchomienie i przeprowadzenie meczu, czyli pojedynczej rozgrywki z wybranym przez siebie botem
- Wyświetlenie szczegółów poszczególnych rozgrywek
- Dodanie, usunięcie i edycję komentarza do gry i do konkursu

7.5. Baza danych

Za obsługę bazy danych odpowiedzialny jest framework Django. Współpracuje z bazami: PostgreSQL, MySQL, SQLite, Microsoft SQL Server oraz Oracle. Domyślną bazą, na której działa serwis jest PostgreSQL. Tabele w bazie danych są tworzone przez Django w sposób automatyczny na podstawie modeli w pliku `models.py`, każdemu modelowi odpowiada jedna tabela w bazie. Oprócz tego istnieje plik `fixture`, do którego można zapisać lub wczytać z niego dane znajdujące się w bazie. Służy on do tworzenia testowej bazy danych. Najważniejsze modele w serwisie to:

- **Game** - zawiera plik z zasadami Gry oraz ścieżkę do programu Sędziego
- **Bot** - reprezentuje bota grającego w konkretną Grę, zawiera m.in. plik źródłowy
- **Contest** - zawiera informacje o Konkursie, a więc Grę, Ranking i listę Botów biorących w nim udział
- **Ranking** - od jego typu zależy w jaki sposób będzie rozgrywany Konkurs, np. "każdy z każdym"
- **BotRanking** - opisuje pozycję Bota w danym Rankingu
- **Match** i **MatchBotResult** - przechowują informację o rozegranym Meczu tzn. które Boty brały w nim udział oraz ich wyniki, wykorzystany czas i zużytą pamięć
- **UserProfile** - najważniejsze dane o użytkowniku tj, zdjęcie, państwo pochodzenia



Rysunek 7.3: Diagram UML bazy danych

7.6. Testowanie

Poprawne działanie serwisu jest testowane przy pomocy standardowych Unit Testów w Django. Framework umożliwia automatyczne testowanie wszystkich warstw aplikacji: obsługi żądań HTTP, walidacji i przetwarzania formularzy, renderowania szablonów. Bardzo proste jest symulowanie żądań, używanie testowych danych oraz weryfikowanie konkretnych widoków i metod.

Aby dodać test jednostkowy wystarczy w pliku tests.py umieszczonym w folderze testo-

wanej aplikacji zdefiniować klasę dziedziczącą po `django.test.TestCase`. Każda z jej metod będzie uruchomiona oddzielnie podczas uruchomienia automatycznych testów. W przypadku gdy dana metoda korzysta z bazy danych, nie używa tej "produkcyjnej", na potrzebę testów tworzona jest nowa, pusta baza. Można jednak wskazać plik zawierający fixtures, w którym jest zapisana testowa baza w formacie json.

Django udostępnia klasę testowego klienta, wykorzystywanego do zaprogramowania interakcji z aplikacją. Przy jego użyciu można symulować żądania GET i POST na konkretny URL i weryfikować odpowiedź (status, zawartość itp.), sprawdzić czy dla danego URL wywoływany jest odpowiedni widok oraz czy odpowiedź jest renderowana przy użyciu właściwego szablonu w odpowiednim środowisku. Jest to możliwe dzięki różnego rodzaju asercjom.

Uruchomienie automatycznych testów jest bardzo proste. Domyślnie używane są wszystkie przypadki testowe z wszystkich aplikacji, ale wolno wyspecyfikować konkretną aplikację lub przypadek testowy. W razie niepowodzenia któregoś z testów wyświetlane są szczegóły błędu.

Testy zawarte w projekcie zostały podzielone na dwie kategorie. Pierwsza, zawarta w klasie *WebPageTest*, to testy symulujące wysyłanie określonych żądań na konkretne adresy url. Wszystkie korzystają z tej samej bazy testowej zapisanej w pliku `test_fixture.json`. Poprawność odpowiedzi jest sprawdzana przy pomocy asercji pod kątem treści, a więc czy zawiera spodziewany tekst. Analizowane są także przekierowania, jeżeli np. edycja danych wymaga zalogowania, to użytkownik powinien być przekierowany na panel logowania, a następnie na odpowiednią stronę.

Na drugą kategorię składają się testy zawarte w klasie *ModelTest*, które sprawdzają poprawność metod modeli wykorzystywanych w serwisie. W tym przypadku nie jest potrzebna baza testowa.

Testy jednostkowe pokrywają kod w całości. Dla każdego prawidłowego adresu url zamieszczonego w pliku `urls.py` istnieje przynajmniej jeden przypadek testowy wysyłający żądanie na ten adres. Podobnie, każda metoda zawarta w pliku `models.py` ma swój odpowiednik w testach.

Rozdział 8

Decyzje technologiczne

Podczas tworzenia projektu Niezbędne było podjęcie kluczowych decyzji definiujących w dużym stopniu kształt dalszej pracy nad serwisem. Najbardziej istotną decyzją był wybór języka programowania. Twórcy zdecydowali się na wykorzystanie Pythona z frameworkiem Django, który w dużym stopniu ułatwia tworzenie i rozwój stron internetowych.

W przypadku backendu również zdecydowano się na użycie Pythona, głównie z uwagi na łatwość rozwoju programów i czytelność kodu w nim napisanego. Zgodnie stwierdziliśmy, iż w przypadku AI-Areny, ze względu na charakter serwisu, wydajność nie jest sprawą kluczową, lecz jedynie poboczną.

8.1. Django

Django jest wysokopoziomowym frameworkiem przeznaczonym do tworzenia aplikacji internetowych. Opiera się na wzorcu projektowym podobnym do MVC nazywanym MVT. Z tego względu aplikację można podzielić na kilka warstw:

- **Model (eng. Model)** - reprezentuje logikę aplikacji. Każdej klasie (zdefiniowanej w pliku `models.py`) odpowiada pewna tabela (lub kilka tabel) w bazie danych
- **Widok (eng. View)** - określa jakie dane będą zaprezentowane użytkownikowi.
- **Szablon (eng. Template)** - obejmuje szablony html. Są one renderowane na podstawie kontekstu obliczonego w warswie widoku.

Oprócz tego framework zapewnia opartą na wyrażeniach regularnych obsługę URL, co pozwala na proste, przejrzyste adresowanie. Django posiada własny, lekki serwer umożliwiający rozwijanie i testowanie aplikacji. Ponadto zajmuje się obsługą bazy danych i systemem cache'owania danych. Do zalet tego frameworku należą też duża skalowalność i wydajność oraz wbudowany panel administracyjny.

Django jest jednym z bardziej popularnych frameworków przeznaczonych do tworzenia aplikacji internetowych. Autorzy projektu zdecydowali się właśnie na ten z kilku względów.

- Rozwiązuje wszystkie podstawowe kwestie związane z projektem, tj. obsługa bazy danych, adresowanie.
- Serwis od strony webowej jest standardową aplikacją, zatem istnieją mechanizmy wbudowane w Django pozwalające w prosty sposób rozwiązać wiele problemów technicznych.

- Ważne było to, że członkowie zespołu znali tę technologię już wcześniej z przedmiotu Aplikacje WWW. Należy zauważyć, iż Django jest proste w użyciu, dzięki maksymalnej automatyzacji wielu standardowych procesów (np. renderowania stron WWW).

Django powstało w celu szybkiego tworzenia standardowych aplikacji internetowych i po doświadczeniach związanych z tworzeniem projektu autorzy w pełni potwierdzają skuteczność tego frameworku.

8.2. Nadzorca

W początkowej fazie projektu autorzy planowali udostępnić użytkownikom możliwość wysyłania programów wieloprotokolowych. Okazało się jednak, że implementacja takiej funkcjonalności stwarza wiele trudnych do rozwiązania problemów w warstwie nadzorca. Głównymi problemami okazały się ograniczenie zużycia czasu i pamięci przez grupę procesów, pomiar zużycia tych zasobów po zakończeniu działania grup procesów, i zabijanie odpowiednich grup procesów na żądanie Sędziego.

Standardowe metody ograniczania zużycia pamięci i czasu przez procesy : ulimit i setrlimit niestety zawodzą. Setrlimit ogranicza jedynie zużycie zasobów przez dany proces, nie ograniczając w żaden sposób jego dzieci. Ulimit ogranicza co prawda wszystkie procesy danego użytkownika, jednak nałożenie osobnych limitów na wszystkie procesy nie jest poprawnym rozwiązaniem, należy bowiem ograniczyć sumaryczne zużycie zasobów przez dany proces i wszystkich jego potomków.

Metody pomiaru udostępniane przez system operacyjny takie jak wait3 lub wait4 również nie znajdują zastosowania w tym przypadku. Co prawda wait3 i wait4 potrafią zebrać informacje o procesie i jego zakończonych dzieciach, jednak informacje o wnukach procesu mogą zostać utracone, jeśli tylko użytkownik serwisu nie będzie sam używał wait3 lub wait4 (czego oczywiście nie możemy założyć ani kontrolować). Należy również pamiętać że informacje udostępniane przez te dwie funkcje są dostępne jedynie po zakończeniu procesu, ich użycie jest więc możliwe jedynie gdy przy pomocy innych mechanizmów zapewnione zostanie, że proces który zużył zbyt dużo zasobów zostanie zakończony.

Inny mechanizm systemowy który był rozważany jako rozwiązanie tych problemów, to grupy procesów. Mechanizm ten pozwalałby między innymi na wysłanie sygnału do procesu i wszystkich jego potomków, jak również prostsze monitorowanie zużycia zasobów systemowych przez grupę procesów. Niestety ponieważ tego samego mechanizmu używa powłoka systemowa, stwarza to złośliwemu użytkownikowi możliwość tworzenia procesów, które zostaną uznane za procesy powłoki i uwolnienia się tym samym spod kontroli.

Na licencji open source dostępne jest rozwiązanie podobnego problemu. Jest to napisany w języku Perl skrypt timeout, stworzony w Institute for System Programming of Russian Academy of Sciences, dostępny przez github (<https://github.com/Buzdygan/ai-arena>). Został on stworzony do mierzenia i ograniczania zużycia zasobów przez eksperymentalne wieloprotokolowe programy, których zachowanie było trudne do przewidzenia. Działanie tego skryptu oparte jest na prostej idei. Skrypt kilkakrotnie w ciągu sekundy sprawdza informacje dotyczące zużycia zasobów przez monitorowany proces i wszystkich jego potomków, i gdy wartość ta przekroczy dozwolony limit zatrzymuje obserwowane procesy. Jednak ponieważ skrypt nie był tworzony do użyciu w sposób jaki jest potrzebny w systemie AI-Arena, brak

mu pewnych funkcjonalności. Szczególnie ważną jest możliwość zatrzymania danego procesu i wszystkich jego potomków na rozkaz Sędziego. Mniej poważną wadą tego rozwiązania jest to, że dokonywane pomiary mogą nie być dokładne, w szczególności niektóre programy mogłyby działać i komunikować się przez ułamek sekundy po przekroczeniu limitu.

Najlepszym rozwiązaniem wydaje się więc zaimplementowanie własnego mechanizmu, opartego na podobnej idei jak wspomniany skrypt `timeout`. Mechanizm ten mógłby być oddzielnym procesem, który na początku swojego działania otrzymuje numery pid obserwowanych procesów i przyznane im limity. Następnie kilkakrotnie w ciągu sekundy na podstawie informacji zawartych w katalogu `proc` mógłby mierzyć sumaryczne zużycie zasobów przez kontrolowane procesy i ich potomków, oraz na życzenie Sędziego przekazać przez nadzorcę zatrzymać określone grupy procesów. Pomiar zużycia zasobów w tym przypadku jest obciążony tym samym błędem co skrypt `timeout`, jest to jednak błąd akceptowalny.

Z powodu wymienionych problemów z kontrolowaniem zużycia zasobów systemowych, jak również potencjalnych problemów z bezpieczeństwem systemu, autorzy zdecydowali, że programy użytkowników i sędziowie nie będą mogli tworzyć nowych procesów.

Ograniczenie zużycia czasu i pamięci zostało zrealizowane przy pomocy funkcji `setrlimit`. Polecenie to pozwala na ustawienie miękkiego i twardego limitu na dostępne procesowi zasoby systemowe, między innymi pamięć wirtualną i czas używany przez proces w trybie użytkownika lub systemowym. W przypadku przekroczenia miękkiego limitu czasowego do procesu wysyłany jest `SIGINT`, po przekroczeniu limitu twardego do procesu wysyłany jest `SIGKILL`. Jasne jest więc że limit który chcemy nałożyć procesowi musi być limitem twardym, gdyż `SIGINT` może zostać obsłużony. Możliwe są również sytuacje w których kernel nie może dostarczyć `SIGKILL` do procesu: gdy proces jest zombie lub gdy czeka w nieprzerywalnym uśpieniu. Jednak w obu przypadkach proces ten nie zużywa już więcej procesora, więc nie stanowi to zagrożenia dla systemu.

Pomiar czasu zużytego przez proces realizowany jest przy użyciu funkcji systemowej `wait4`. Potrafi ona czekać na zakończonego potomka o danym pid, jednocześnie zwracając pewne informacje dotyczące zużytych przez niego zasobów w trakcie jego działania. Wśród tych informacji znajdują się ilość czasu jaką proces spędził w trybie użytkownika lub w trybie systemowym, mierzone z dokładnością co do milisekundy.

8.3. Bezpieczeństwo Systemu

Ponieważ główna funkcjonalność systemu jest oparta o kompilację i uruchamianie kodów źródłowych nieznanego pochodzenia, w naturalny sposób największym zagrożeniem dla systemu są złośliwe kody wysyłane przez użytkowników. Zabezpieczenie przed takimi kodami można rozdzielić na dwa główne podproblemy: zabronienie procesom uruchamiającym programy graczy używania funkcji systemowych które nie są niezbędne dla programów grających w gry, oraz umieszczenie tych procesów w sandboxie, aby złośliwy kod nie miał możliwości odczytu i modyfikacji danych na serwerze.

Autorzy systemu rozważali trzy różne rozwiązania pierwszego podproblemu: `ptrace`, `LD_PRELOAD` i `systrace`.

`Ptrace` jest unixowym narzędziem.

Rozdział 9

Dokumentacja użytkowa

Najważniejszym dokumentem określającym pożądany kształt serwisu jest dokumentacja przypadków użycia. Opisuje on sposób interakcji końcowego użytkownika z całym systemem. W przypadku platformy AI-Arena interakcja ta odbywa się za pomocą przeglądarki internetowej.

Z punktu widzenia użytkownika AI-Arena jest zwykłą stroną internetową. Największy wpływ na jego końcowy wygląd mają więc arkusze stylów, oraz w pewnym stopniu skrypty JavaScript. Arkusze stylów pochodzą z dwóch źródeł: wykorzystany został open source'owy projekt należący do twittera o nazwie bootstrap (<http://twitter.github.com/bootstrap/>). Projekt ten (opublikowany na licencji Apache v.2.0) stanowi dobrą podstawę wyglądu strony. Drugim źródłem arkuszy stylów są autorzy serwisu, którzy nadali stronie ostateczny wygląd.

Użytkownicy serwisu mają zróżnicowane prawa. Najbardziej ograniczone możliwości ma użytkownik niezalogowany, najszersze zaś - naturalnie Administrator strony. Ponadto w zależności od sytuacji Użytkownik (który się poprawnie zalogował) może mieć prawa Moderatora gry lub konkursu, co daje mu możliwość edycji m.in. zasad, terminów czy komentarzy wypowiadanych przez innych Użytkowników. Należy jednak zauważyć, iż w przypadku nadużyć Administrator strony może odebrać Moderatorowi jego specjalne uprawnienia.

9.1. Przypadki użycia systemu

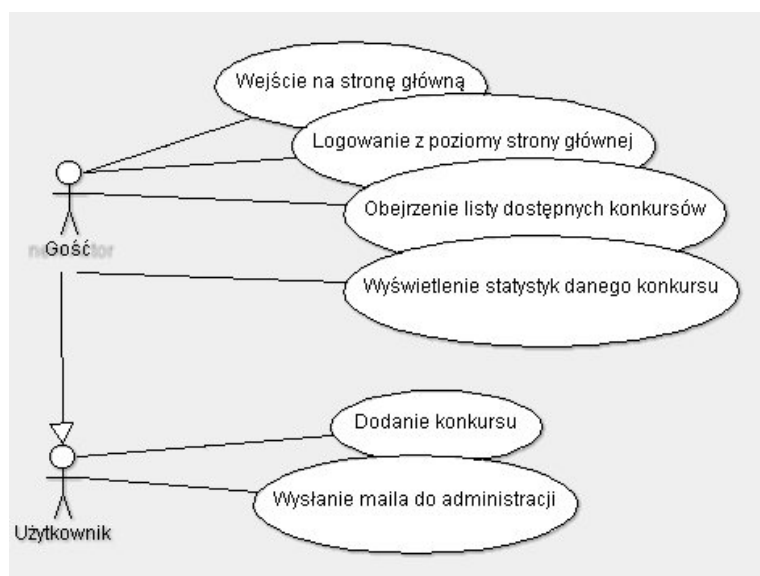
9.1.1. Z udziałem użytkownika niezalogowanego (Gościa)

Użytkownik niezalogowany ma bardzo ograniczone prawa używania serwisu. Praktycznie jedyną dozwoloną aktywnością jest rejestracja w serwisie lub próba zalogowania się do utworzonego wcześniej konta.

Rejestracja w systemie

Aby zarejestrować się w systemie należy wykonać kilka kroków. Na początku należy kliknąć w napis "Register" zlokalizowany po prawej stronie na górze.

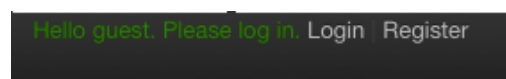
Następnie system przekieruje użytkownika do strony, gdzie będzie mógł wybrać swój login oraz hasło. Należy zwrócić uwagę, iż login musi być unikalny, może zawierać wyłącznie małe



Rysunek 9.1: Podstawowe użycie systemu

i duże litery, cyfry oraz znak '_'. W przeciwnym wypadku pojawi się komunikat o błędzie.

Po prawidłowym wykonaniu tego kroku użytkownik zostanie przeniesiony na swoją stronę profilową. Może stąd zarządzać swoim kontem. Od tej chwili jest również zalogowany w serwisie, co znaczy, że może m. in. wysłać swoje boty do gier lub wykonywać inne akcje dostępne wyłącznie dla zalogowanych użytkowników.



Rysunek 9.2: Pasek logowania i rejestracji do serwisu

Logowanie

Logowanie jest standardową aktywnością, którą się spotyka na wszystkich forach i wielu serwisach. Przycisk logowania znajduje się tuż obok przycisku rejestracji. Aby się zalogować należy kliknąć w ten przycisk a następnie wypełnić pola formularza, który się pojawi na ekranie. Jeśli dane będą zgodne z tymi zapisanymi w bazie danych, system przekieruje użytkownika na jego stronę profilową.

9.1.2. Z udziałem użytkownika zalogowanego

Wylogowanie się z serwisu

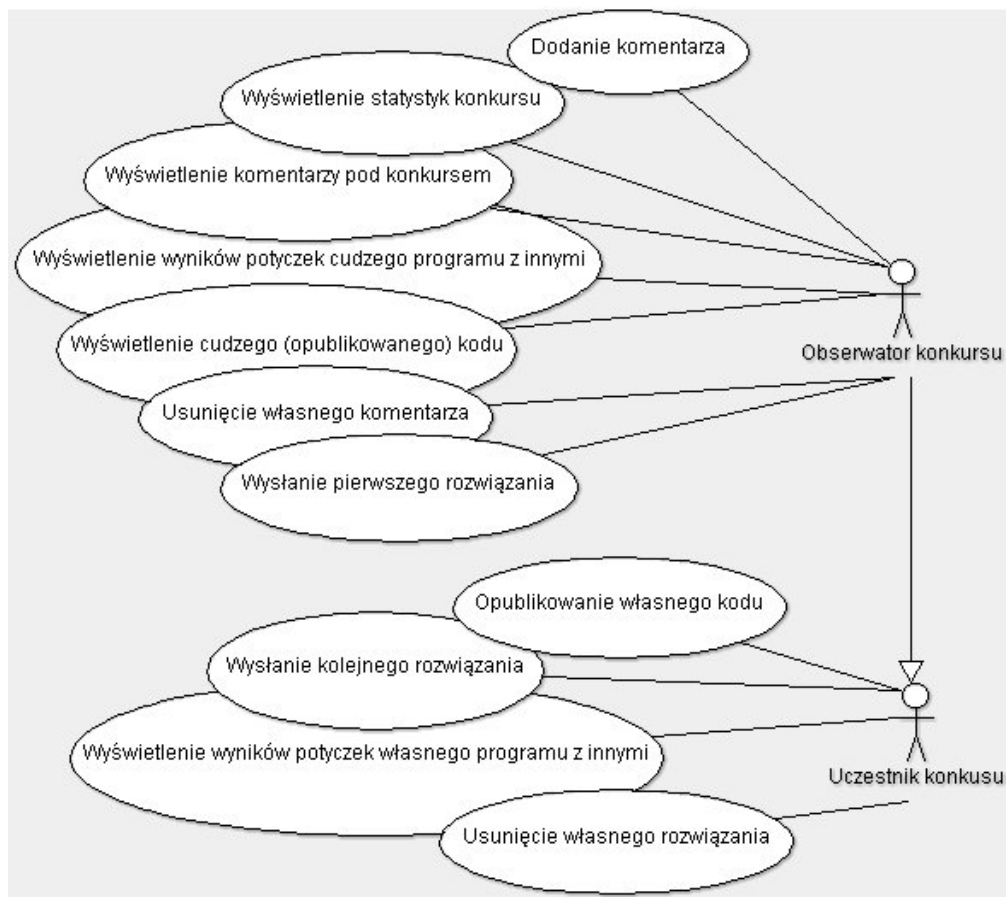
Wylogowanie jest kolejną standardową aktywnością udostępnianą przez wiele serwisów. Aby wylogować się z serwisu należy kliknąć przycisk "Logout", który znajduje się w prawym, górnym rogu strony. Zastępuje on przycisk "Login", który widoczny jest dla niezalogowanych użytkowników. Po kliknięciu tego przycisku użytkownik jest przekierowywany na stronę główną serwisu.

Otwarcie swojej strony profilowej

Aby otworzyć swoją stronę profilową użytkownik musi kliknąć w link znajdujący się pod jego loginem. Loginy użytkowników pojawiają się w różnych miejscach systemu i kliknięcie w dowolny z nich powoduje otwarcie strony profilowej danego użytkownika. W szczególności po kliknięciu na własny login pojawi się własna strona użytkownika. Istnieje również tzw. "stały link do profilu" znajdujący się na górze strony w prawym rogu. Po kliknięciu w znajdujący się tam login użytkownika (zaznaczonego innym kolorem niż reszta napisu) system wyświetla stronę profilową aktualnie zalogowanego użytkownika.

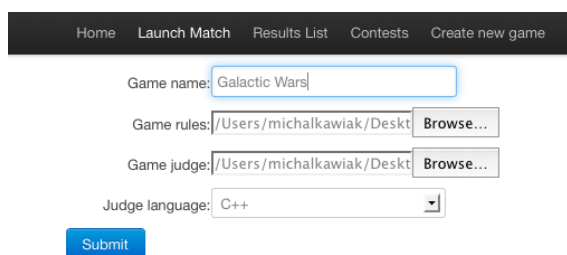
Edycja profilu użytkownika

Gdy użytkownik znajduje się na swojej stronie profilowej ma dostęp do kilku odnośników. Jednym z nich jest odnośnik do strony konfiguracyjnej profilu. Aby ją otworzyć należy kliknąć w link "Edit profile" znajdujący się pod zdjęciem profilowym. W oknie edycji profilu pojawia się formularz zawierający kilka pól, m.in. zdjęcie, kraj, szkoła/universytet czy zainteresowania. Żadne z nich nie jest obowiązkowe. Użytkownik zatwierdza zmiany klikając na przycisk "Update" znajdujący się na samym dole formularza.



Rysunek 9.3: Przypadki użycia związane z konkursami

Utworzenie nowej gry lub konkursu



Rysunek 9.4: Formularz do tworzenia nowej gry

nikacie nie narusza reguł gry (przykładowo, że gracz grający pionami czarnymi nie ruszył się białym pionem). Gdy upewni się, że otrzymany komunikat jest poprawny wysyła na swoje standardowe wyjście komunikat zgodnie z formatem opisanym w regułach gry.

Plik źródłowy Sędziego jest dostępny dla wszystkich zalogowanych użytkowników serwisu. Ma to zapobiec wysyłaniu "stronniczych" sędziów oraz łatwieszemu ujawnieniu pewnych "ukrytych" wad protokołu. Administratorzy zastrzegają sobie prawo do usunięcia lub poprawienia wadliwych sędziów.

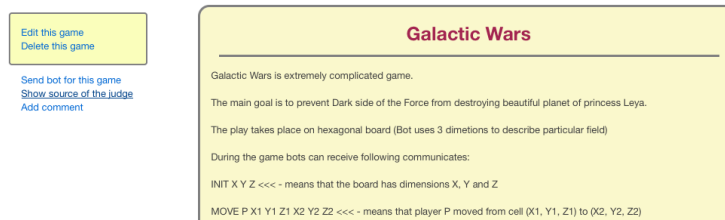
Plik z zasadami rozgrywki musi zawierać ogólny opis gry (np. historię wprowadzającą) oraz szczegółowy opis i znaczenie komunikatów. W szczególności muszą być zawarte informacje takie jak dokładny format komunikatu, dozwolone odpowiedzi na każdy rodzaj komunikatu czy komunikaty informujące o błędzie. Zostało przyjęte założenie, że Bot, który wyśle komunikat o innej, niż zdefiniowanej formie automatycznie przegrywa daną grę i jest przerywany przez Nadzorcę.

Aby utworzyć konkurs nie jest potrzebne wcześniejsze przygotowywanie plików, jednak musi istnieć gra, na bazie której chcemy utworzyć nowy konkurs.

Gdy użytkownik przygotował już powyższe pliki powinien, w celu utworzenia nowej gry, kliknąć w link "Create New Game" znajdujący się na górnym pasku strony. Otworzy się formularz zawierający pola, do których należy wprowadzić informacje nt. tworzonej gry (między innymi 2 wymienione wcześniej

Tworzenie gry (konkursu) jest jedną z centralnych aktywności w serwisie. Aby utworzyć grę, użytkownik musi mieć wcześniej przygotowany plik źródłowy Sędziego oraz plik z zasadami rozgrywki.

Sędzia jest programem, który nadzoruje przebieg gry. Powinien być napisany w jednym z języków: C, C++, Java, Python. Otrzymuje on komunikaty od Nadzorcy, który z kolei zczytuje je ze standardowego wejścia Botów. Sędzia musi dokonać walidacji komunikatu, tj. upewnić się, iż format komunikatu jest zgodny z protokołem opisanym w zasadach gry oraz, że ruch zawarty w komunikacie

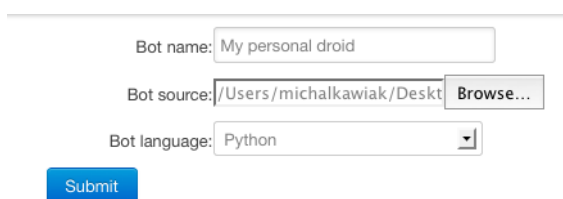


Rysunek 9.5: Szczegóły gry

pliki czy nazwa gry). Warto zwrócić uwagę, iż nazwa gry musi być unikalna, gdyż definiuje ona ścieżkę do poszczególnych plików na dysku.

Gdy użytkownik wypełni formularz i zatwierdzi przyciskiem "Submit" na dole formularza zostanie przeniesiony na stronę zawierającą detale właśnie utworzonej gry. Autor staje się automatycznie moderatorem gry i ma prawo m. in. do edycji czy usunięcia gry lub edycji i usuwania komentarzy użytkowników.

Wysyłanie bota do gry lub konkursu



Rysunek 9.6: Formularz do wysyłania bota

Aby wysłać bota do gry (konkursu) należy w pierwszej kolejności przejść na stronę zawierającą szczegółowy opis gry. Można to zrobić klikając na przycisk "Games list" ("Contests") znajdujący się na górze strony, a następnie wybrać grę (konkurs) z wyświetlonej listy.

Gdy znajdujemy się już na stronie z detalami należy, w celu zgłoszenia bota, kliknąć na link "Send

bot for this game" (lub "Add contestant" w przypadku konkursu) i wypełnić formularz wyświetlony przez system. Podobnie, jak w przypadku tworzenia gry należy mieć uprzednio przygotowany plik zawierający kod źródłowy bota. Bot komunikuje się z sędzią za pomocą komunikatów opisanych w zasadach gry wypisywanych i czytywanych ze standardowego wejścia i wyjścia. W przypadku, gdy bot wyśle źle sformatowany lub błędny komunikat - automatycznie przegrywa rozgrywkę dostając za nią możliwe minimum punktów.

Wyświetlenie kodu źródłowego Sędziego

```
1 /* quick and dirty hack to grab all credentials in the cred hash table
2  * from kernel via sysctl.
3  * sysctl is only defined if xnu is built with DEBUG_CRED defined.
4  */
5
6 #include <stdio.h>
7 #include <stdlib.h>
8 #include <fcntl.h>
9 #include <limits.h>
10 #include <string.h>
11 #include <errno.h>
12 #include <unistd.h>
13 #include <sys/stat.h>
14 #include <sys/types.h>
15 #include <sys/sysctl.h>
16 #include <bsm/audit.h>
17
18 /* badi this is replicated in kern_credential.c, make sure they stay in sync!
19  * Or better yet have a common header file?
20  */
21 struct debug_ucred {
22     uint32_t credp;
23     uint32_t cr_ref; /* reference count */
24     uid_t cr_uid; /* effective user id */
25 }
```

Rysunek 9.7: Sposób wyświetlania kodu źródłowego

W celu wyświetlenia kodu źródłowego Sędziego należy przejść na stronę gry zawierającą szczegółowy jej opis. Potem, po kliknięciu w link "Show source of the judge" system przekieruje nas na stronę, gdzie wyświetlony zostanie kod źródłowy Sędziego.

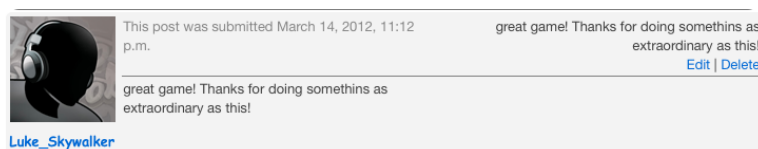
Możliwe jest również wyświetlenie kodu w postaci surowej, tj. bez numerów linii i dodatkowego formatowania tekstu. Aby wyświetlić surowy kod należy kliknąć w przycisk "Raw" znajdujący się w prawym górnym rogu wyświetlanego kodu. Dostępna jest ponadto opcja pobrania kodu źródłowego Sędziego w postaci pliku

tekstowego. Aby pobrać kod źródłowy użytkownik powinien kliknąć w przycisk "Download" znajdujący się na prawo od przycisku "Raw".

Dodanie komentarza pod grą lub konkursem

Aby dodać komentarz do gry (konkursu) należy znajdować się na stronie ze szczegółowym opisem danej gry (konkursu). Następnie kliknąć w link "Add comment" i wypełnić formularz wyświetlony przez system. Komunikat ten pojawi się pod opisem zasad gry (bez konieczności zatwierdzania go przez Moderatora). Moderator ma jednak prawo edytować lub nawet usunąć ten komentarz w późniejszym terminie.

Zwróćmy uwagę na fakt, iż na późniejszym etapie życia serwisu liczba umieszczanych komentarzy może być znacząca. Jakakolwiek forma automatycznej notyfikacji mogłaby być uciążliwa dla Moderatorów. Z tego powodu autorzy serwisu zdecydowali się nie umieszczać mechanizmu automatycznego powiadamiania w serwisie.

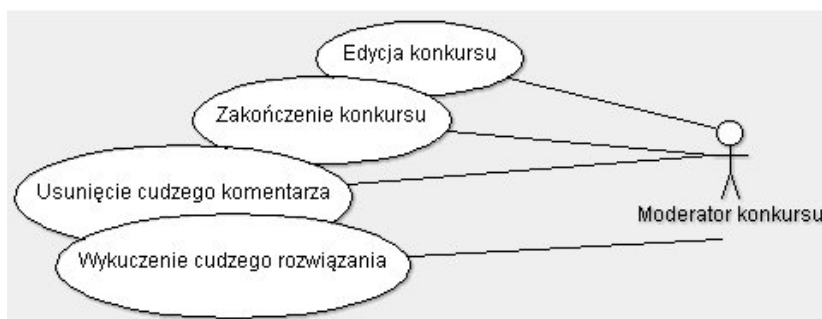


Należy zwrócić uwagę, iż w przypadku naruszania zasad publicznego wypowiedzania się (treści wulgarne, obraźliwe, naruszające prywatność itp.) moderator konkursu ma prawo do edycji

Rysunek 9.8: Komentarz umieszczony pod grą lub usunięcia komentarza. Takie samo prawa ma również autor komentarza.

9.1.3. Z udziałem moderatora konkursu lub gry

Moderator pełni specjalną funkcję. Czuwa on nad prawidłowym i kulturalnym przebiegiem konkursu. W serwisie AI-Arena moderatorzy delegowani są do poszczególnych gier i konkursów. Naturalnie dany użytkownik może być moderatorem wielu gier jednocześnie.



Rysunek 9.9: Użycie systemu przez Moderatora

Usunięcie komentarza do gry lub konkursu

Żeby móc usunąć komentarz użytkownik musi spełniać (minimum) jeden z poniższych warunków:

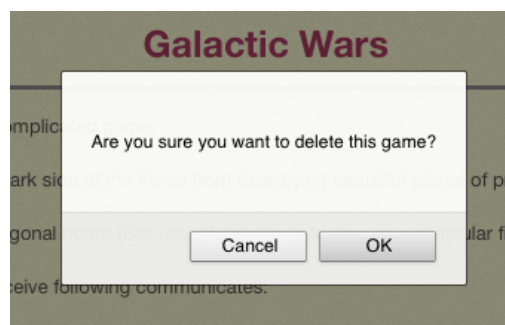
1. Być autorem komentarza
2. Być Administratorem strony lub Moderatorem gry (konkursu), pod którym został ten komentarz dodany

Jeśli wymaganie wstępne jest spełnione, wystarczy kliknąć w przycisk "Delete" znajdujący się w prawym górnym rogu komentarza. Pojawi się wtedy okienko z prośbą o potwierdzenie swojej decyzji. Należy pamiętać, że usuniętego komentarza nie można przywrócić! Wciśnięcie przycisku "OK" na pojawiającym się okienku spowoduje usunięcie komentarza i powrót do strony z detalami gry (konkursu)

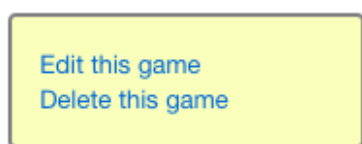
Usunięcie gry lub konkursu

Aby móc usunąć grę (konkurs) trzeba być Administratorem strony lub Moderatorem danej gry (konkursu):

W celu usunięcia gry należy znajdować się na stronie zawierającej szczegółowy opis gry. Następnie kliknąć w znajdujący się po lewej stronie link "Delete this game". Spowoduje to wyskoczenie okienka z prośbą o potwierdzenie swojej decyzji. Pamiętajmy, iż jest to decyzja nieodwracalna! Potwierdzenie wykonania operacji spowoduje usunięcie gry z listy dostępnych i przeniesienie na stronę główną serwisu. System usunie wszelkie dane związane z daną grą.



Edycja gry lub konkursu



[Send bot for this game](#)
[Show source of the judge](#)
[Add comment](#)

Możliwa jest edycja konkursu. Aby tego dokonać trzeba być administratorem lub moderatorem gry. Należy kliknąć w link "Edit game". System przeniesie nas do okna, w którym wyświetli się formularz do edycji gry. Po wprowadzeniu zmian należy zatwierdzić przyciskiem "Save".

Należy pamiętać, iż nazwa gry określa fizyczną ścieżkę dostępu. Oznacza to, że jeśli użytkownik będzie chciał zmienić nazwę gry, system przeniesie wszystkie pliki związane z grą do nowych lokalizacji. Użytkownik musi jednak pamiętać, aby nowa nazwa gry była unikalna. W przeciwnym razie system odmówi zmiany nazwy i próba edycji zakończy się porażką.

Edycja komentarza do gry lub konkursu

Edycja komentarza jest podobna do dodawania nowego komentarza. Aby edytować istniejący komentarz należy kliknąć w link "Edit" znajdujący się w górnej części komentarza. Pojawi się wtedy formularz analogiczny do tego, który się pojawia przy dodawaniu komentarza. Po zatwierdzeniu dokonywanych zmian poprzez naciśnięcie przycisku "Save" zostaniemy przekierowani do strony ze szczegółowymi informacjami o grze. Zauważmy, że edytowany komentarz jest już wyświetlany w zmienionej formie.

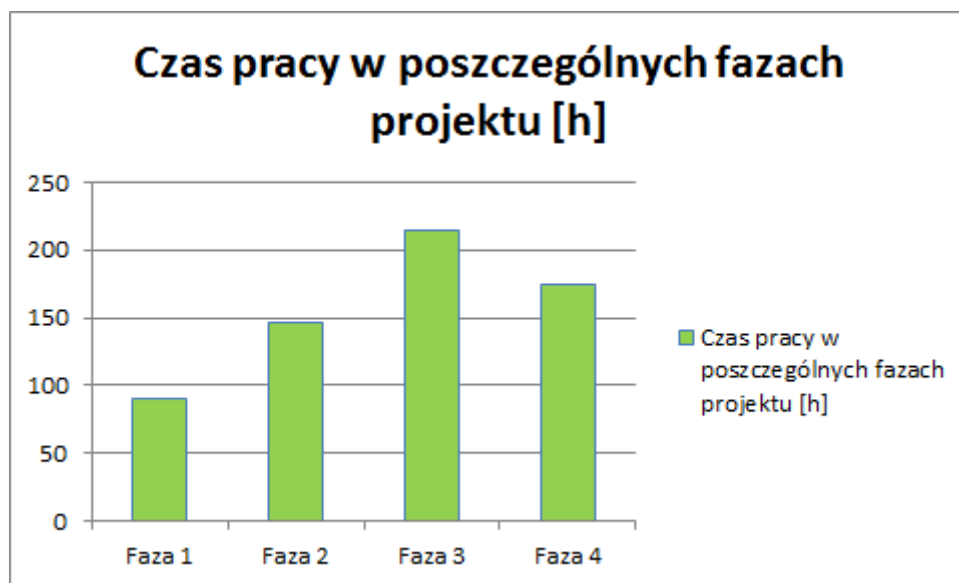
Rozdział 10

Harmonogram prac

Praca nad projektem została podzielona na cztery kluczowe okresy:

- Stworzenie działającego prototypu - termin wykonania: **do 30 listopada 2011r.**
- Wykonanie wersji minimalistycznej serwisu - termin wykonania: **do 30 stycznia 2012 r.**
- Wykonanie wersji rozszerzonej serwisu - termin wykonania: **do 4 kwietnia 2012 r.**
- Dopracowywanie interfejsu i ostatnie testy użytkownika - termin wykonania: **do 15 maja 2012 r.**

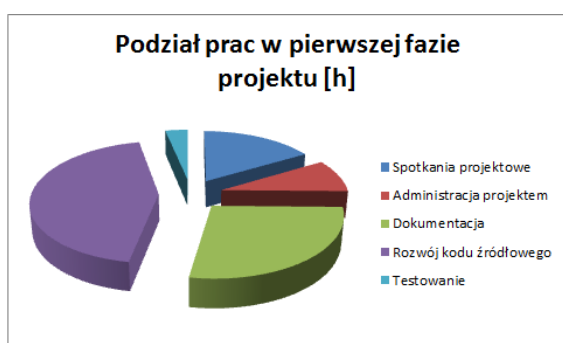
Zauważmy, że taki plan prac umożliwiał, w wypadku nieporzewidzianych okoliczności, możliwość zrealizowania ograniczonej funkcjonalności. Okazało się jednak, że opcja ta nie była potrzebna, gdyż praca postępowała zgodnie z harmonogramem.



Rysunek 10.1: Podział pracy w kolejnych fazach rozwoju projektu

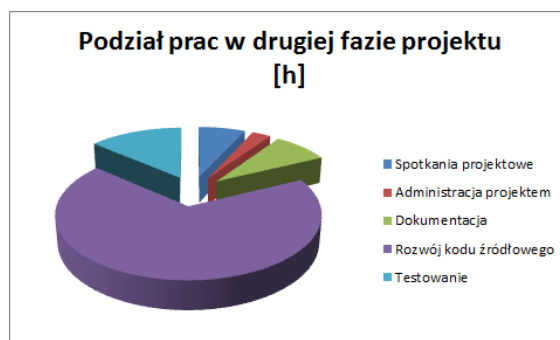
Wykresy prezentowane w dalszej części rozdziału obrazują podział prac między poszczególne zagadnienia związane z rozwojem projektu takie jak spotkania projektowe, administracja projektem, tworzenie dokumentacji i automatycznej dokumentacji, rozwój kodu czy testowanie. Należy pamiętać, iż wielkości prezentowane na wykresach są jedynie orientacyjne, jednak dobrze oddają ilość pracy, jaka została przez nas włożona w projekt.

Spotkania projektowe oznaczają spotkania zespołu, w czasie których omawiana była wizja projektu czy bieżący podział prac. Administracja uwzględnia takie zagadnienia jak obsługa systemu Redmine, obsługa repozytorium czy wdrożenie projektu na serwer. Obejmuje również obsługę majowego konkursu przeprowadzonego pod patronatem Codility. Tworzenie dokumentacji i automatycznej dokumentacji opisuje czas potrzebny na stworzenie wszystkich dokumentów, włącznie z pracą licencjacką, oraz dodaniem docstringów służącym do generowania automatycznej dokumentacji. Testowanie zaś oznacza czas przeznaczony na napisanie unit testów oraz przeprowadzenie test-case'ów. Obejmuje ponadto zagadnienie kontroli kodu przez innych członków zespołu.



W pierwszej fazie projektu skupiono się na kluczowych zagadnieniach mających wpływ na ostateczny kształt serwisu. Powstały wtedy m.in. takie dokumenty jak Software Development Plan czy przypadki użycia. Została uzgodniona wspólna wizja i opracowane zostały szczegóły techniczne. Zaprojektowany został ponadto model interakcji z użytkownikiem końcowym. Ujrzała światło dzienne pierwsza wersja nadzorcy oraz powstał szkielet projektu utworzony w Django.

W efekcie tych prac, przed końcem listopada można było zaprezentować działający przykład gry dwuosobowej nadzorowanej przed Nadzorcą. Gra ta jednak była uruchamiana z linii poleceń. Warto jednak wspomnieć, iż model komunikacji między uczestnikami (botami i sędzią) był bliski końcowego, tj. programy porozumiewały się korzystając ze standardowego wejścia i wyjścia.

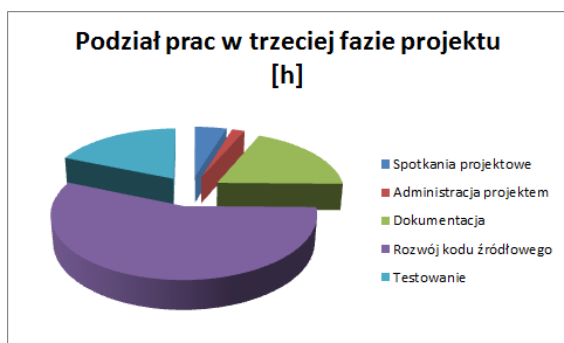


Druga faza projektu zaowocowała powstaniem interfejsu graficznego. Możliwe stało się logowanie i rejestracja na stronie, oraz podstawowa działalność związana z użytkowaniem serwisu. Dało się m.in. stworzyć no-

wą grę czy wysłać bota grającego w daną grę (i realizującego protokół określony w regułach gry).

Udostępniona została ponadto możliwość uruchomienia meczu pomiędzy dowolnymi dwoma botami grającymi w tą samą grę. Wyniki takich meczów były następnie zapisywane w bazie danych. Zostały również poczynione pierwsze kroki w celu zapewnienia bezpieczeństwa na stronie. Dodano podstawową obsługę błędów zabezpieczającą przed najprostszymi typami ataków.

Dodatkowym rezultatem tej części projektu było wprowadzenie do projektu AI-Arena modułu Gearman - zewnętrznego modułu odpowiedzialnego za kolejkowanie zleconych zadań. W czasie tej fazy została dodana również funkcjonalność automatycznej generacji dokumentacji.



Podczas trzeciej fazy projektu został znacznie rozwinięty interfejs użytkownika. Zostały dodane funkcjonalności dodatkowe, zbędne z punktu widzenia funkcjonowania serwisu, jednak ułatwiające jego użycie. Powstały moduły odpowiedzialne za wyświetlanie kodu źródłowego, dodawanie konkursów, profile użytkownika i preferencje, ranking czy komentarze. Strona internetowa zyskała także nowy wygląd dzięki arkuszom stylów. W niektórych miejscach zostały dodane również skrypty JavaScript, sprawiające, że interakcja z użytkownikiem stała się bardziej dynamiczna.

W międzyczasie trwały prace również nad warstwą Nadzorcy. Dodane zostały skrypty umożliwiające automatyczną kompilację gier i botów wysłanych za pośrednictwem serwisu webowego. Dodatkowo umożliwiono Nadzorcy przejęcie kontroli nad procesem (w celu jego zakończenia), który przekroczył swój limit czasu bądź pamięci lub będzie postępował niezgodnie z regułami gry.

Udostępniona została botom również możliwość zapisywania logów na standardowy strumień błędów. Komunikaty wypisane na strumień błędów będą przekazywane botom po zakończeniu danej rozgrywki.

Trzecia faza projektu zakończona została włożeniem systemu na udostępniony gościnnie, przez Codility, serwer. Zostaliśmy przy tym poproszeni o przeprowadzenie w połowie maja konkursu, pod patronatem Codility, w którym platforma AI-Arena miałaby odgrywać znaczącą rolę.

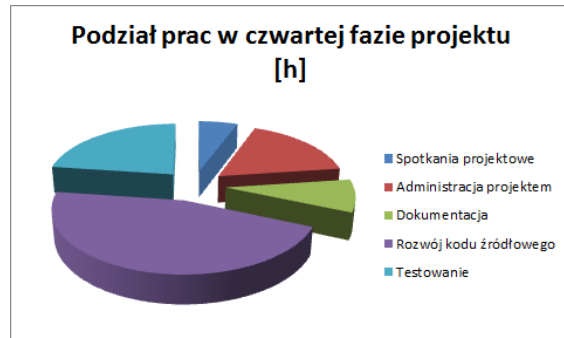
Istotnym rezultatem tej części projektu było również napisanie znaczącej części pracy licencjackiej. Późniejsze zmiany ograniczały się do wprowadzania mniejszych zmian styli-

stycznych, bądź poprawienia ogólnego układu pracy.

Finalna, czwarta faza projektu obfitowała w wiele poprawek kodu, mających na celu usprawnienie i poprawę interakcji z użytkownikiem. Nie zostały dodane do frontendu żadne znaczące funkcjonalności. Udoskonalona została natomiast obsługa błędów. Duży nacisk położono również na bezpieczeństwo środowiska, w którym sędzia i boty są uruchamiane. Do zapewnienia bezpieczeństwa wykorzystano tzw. Sandboxa - mechanizm pozwalający na odizolowanie środowiska systemu od tymczasowego środowiska programu.

Ostateczną formę przybrała również dokumentacja kodu. Została dołączona do repozytorium razem z pozostałą częścią projektu. Można ją w wygodny sposób przeglądać z wykorzystaniem przeglądarki internetowej.

Przeprowadzony został szereg testów: zarówno testów sprawdzających scenariusze użycia systemu, jak i sprawdzające przypadki niewłaściwego jego użycia. Kluczowym testem okazał się konkurs zorganizowany na prośbę Codility. Okazało się, że « Do uzupełnienia po tym konkursie »



Rozdział 11

Przyszłość projektu

Projekt powinien rozwijać się jako niezależny serwis internetowy. Jest on projektem open-source’owym udostępnionym na licencji GPL, czyli osoby zainteresowane rozwijaniem go, mogą łatwo uzyskać prawa do modyfikacji kodu. Licencja GPL jest najbardziej popularną licencją wolnego oprogramowania, daje ona użytkownikom następujące cztery wolności:

- wolność uruchomienia programu w dowolnym celu
- wolność analizowania jak program działa i dostosowywania go do swoich potrzeb
- wolność rozpowszechniania niezmodyfikowanej kopii programu
- wolność udoskonalania programu i publicznego rozpowszechniania własnych ulepszeń

Autorzy serwisu mają nadzieję, że użycie licencji GPL pozwoli serwisowi AI-Arena rozwijać się przy pomocy społeczności programistów sztucznej inteligencji gdyby twórcy przestali mieć możliwość dalszego zajmowania się serwisem.

Należy przy tym dodać, iż licencja GPL zakłada, iż każdy program wykorzystujący kod udostępniony na zasadzie tej licencji również będzie udostępniony na tych samych zasadach. Ogranicza to możliwość wykorzystania kodu projektu w zastosowaniach wymagających tajności kodu. Autorzy pragną w tym miejscu zachęcić wszystkich zainteresowanych dalszym rozwojem do kontaktu z nami oraz do odwiedzenia repozytorium zawierającego najbardziej aktualną wersję (<https://github.com/Buzdygan/ai-arena/>). Repozytorium to zawiera również powstałą dokumentację (również generowaną automatycznie). Ponieważ na chwilę obecną liczymy bardziej na polskich developerów, większość dokumentów jest dostępna tylko w języku polskim. Wyjątkiem jest dokumentacja generowana automatycznie napisana w języku angielskim.

Ambicją twórców serwisu jest uczynienie go centrum rozwoju dziedziny sztucznej inteligencji związanej z tworzeniem inteligentnych botów, działających w warunkach rywalizacji. Swoboda w dodawaniu własnych gier/wyzwań umożliwi rozwijanie serwisu przez jego użytkowników. Istotnym celem jest stworzenie społeczności programistów/badaczy, którzy mogliby wymieniać się pomysłami na temat rozwiązań na łamach serwisu.

Serwis może być wykorzystany również przez firmy szukające rozwiązań skomplikowanych problemów optymalizacyjnych ¹. Wystarczy, że stworzą odpowiednią grę i zorganizują kon-

¹Zauważmy, że w tym przypadku licencja GPL nie ogranicza firm. Rozwiązanie zgłoszonego problemu są bowiem własnością Użytkownika, który je wysłał w ramach normalnego użycia systemu (nie zaś jako

kurs dla użytkowników serwisu. Przykładem platformy organizującej podobne konkursy może być tunedit.com.

Pierwszą okazją do wykorzystania AI-Arena będzie konkurs organizowany we współpracy z firmą Codility. Odbędzie się on przy okazji finałów mistrzostw świata w programowaniu zespołowym organizowanego w maju 2012 roku w Warszawie. Będzie to konkurs programów walczących w grę przygotowaną przez Codility. Za kontrolowanie rozwiązań uczestników, rozgrywanie meczy i opracowanie rankingu będzie odpowiedzialne AI-Arena.

wykorzystanie w ramach swojego projektu). Nie jest on zatem zobowiązany do publikowania swojego kodu i może z nim zrobić co chce, w szczególności może odsprzedać go zainteresowanej firmie.

Rozdział 12

Podział prac

12.1. Jakub Tlałka

- Rola w zespole - Lider projektu
 - Ustalanie terminów dodatkowych spotkań
 - Koordynowanie zadań w platformie Redmine
 - Prezentacja postępów prac i planów na najbliższy okres, na spotkaniach z prowadzącym zajęcia
- Kod źródłowy
 - Modele w bazie danych
 - Ekran uruchamiania pojedynczej rozgrywki
 - Generowanie rankingu do konkursu
 - Wyświetlanie informacji o konkursie wraz z rankingiem
 - Testy interfejsu graficznego
 - Automatyczna dokumentacja
- Praca licencjacka
 - Rozdział 1: Wprowadzenie
 - Rozdział 2: Metodologia Pracy
 - Rozdział 4: Podstawowe pojęcia
 - Rozdział 5: Zastosowania w nauce i biznesie
 - Rozdział 6: Podobne platformy
 - Dodatek A: przykładowe gry
- Dokumentacja
 - Plan architektury serwisu
 - Model bazy danych
- Inne

- Ustalenie współpracy z serwisem Codility - wspólne zorganizowanie konkursu programów walczących przy okazji akademickich mistrzostw świata w programowaniu zespołowym
- Wdrożenie serwisu AI-Arena na serwerze Codility
- Usuwanie błędów
- Kontrola przesyłanego kodu

12.2. Michał Kawiak

- Rola w zespole - Zastępca lidera projektu
- Kod źródłowy
 - Moduł logowania i rejestracji
 - Profil użytkownika
 - Tworzenie nowej gry
 - Wyświetlanie szczegółów gry
 - Dodawanie komentarzy do gier
 - Dodawanie komentarzy do konkursów
 - Usuwanie i edycja gier
 - Usuwanie i edycja konkursów
 - Wyświetlanie kodu źródłowego Sędziego
 - Wysyłanie bota do gry z menu gry
 - Edycja profilu użytkownika
 - Dodanie podstawowej funkcjonalności JS do serwisu
 - Zapewnienie części obsługi błędów
 - Dodanie ról użytkowników: Moderator gry i konkursu
 - Kompilacja przesłanych programów z użyciem skryptów napisanych przez Szymona Majewskiego
- Praca licencjacka
 - Rozdział 9: Dokumentacja użytkowa i opis implementacji
 - Rozdział 11: Harmonogram prac
 - Rozdział 3: Wymagania projektowe
 - Rozdział 7.4: Serwis webowy
 - Część rozdziału 2: Metodologia pracy - dotycząca opisu procesu śledzenia zmian
 - Mniejsze poprawki w pracy
- Dokumentacja
 - Dokumentacja przypadków użycia
 - Diagram przypadków użycia
 - Software Development Plan

- Pierwsza wersja wizji serwisu
- Inne
 - Pomoc Jakubowi Tlałce w koordynacji projektu - głównie w zakresie GUI
 - Usuwanie powstających na bieżąco błędów w kodzie
 - Kontrola przesłanego kodu

12.3. Szymon Majewski

- Rola w zespole - Deweloper backendu
- Kod źródłowy
 - Przesyłanie komunikatów
 - Przeprowadzanie rozgrywki
 - Zbieranie logów
 - Ograniczenie zużycia zasobów przez procesy
 - Pomiar zużycia zasobów przez procesy
 - Automacyjny makefile
- Praca licencjacka
 - Początkowy szkielet pracy
 - Rozdziały dotyczące nadzorcy 7.2 i 8.1
 - Fragmenty rozdziałów 1 i 2
- Dokumentacja
- Inne

12.4. Karol Żebrowski

- Rola w zespole - Tester/Deweloper
- Kod źródłowy
 - Wykorzystanie frameworku Gearman
 - Automatyczne testy aplikacji
 - Testowanie z poziomu GUI
 - Usuwanie drobnych błędów
- Praca licencjacka
 - Podrozdział 7.3 Scheduler
 - Podrozdział 7.5 Baza danych
 - Podrozdział 7.6 Testowanie
 - Podrozdział 8.1 Django

- Fragmenty rozdziału 5
 - Drobne poprawki stylistyczne
- Dokumentacja
 - Wizja serwisu
 - Komunikacja ze Schedulerem Partii
- Inne
 - Research na temat Gearmana
 - Przygotowanie i poprowadzenie prezentacji na zajęciach ZPP pt.”Django - porównanie z innymi web frameworkami”

Rozdział 13

Podsumowanie

Praca nad serwisem AI-Arena była ciekawym, ale wymagającym doświadczeniem. Słuszną decyzją było rozdzielenie struktury projektu na niezależne części i zgodny z nimi podział pracy w zespole. Największym wyzwaniem okazało się zaprojektowanie systemu rozgrywania meczy, tak by niemożliwe były żadne nadużycia ze strony użytkowników, a jednocześnie pozwalał on na dużą swobodę dodawania różnorodnych gier. Za bardzo pozytywny fakt można uznać szybkie zainteresowanie i zapotrzebowanie na wykorzystanie AI-Arena przy okazji Mistrzostw Świata w Programowaniu Zespołowym.

Postępy w pracach były regularne dzięki co tygodniowym spotkaniom na zajęciach z prowadzącym. Wyznaczanie krótkoterminowych celów zapobiegło opóźnieniom względem planowanego przyrostu. Równie dobrym rozwiązaniem były spotkania w obrębie zespołu, na których kontrolowany był postęp prac w danym tygodniu. Najczęściej używanym przez nas narzędziem do koordynowania pracy okazały się Google Documents. Dodatkowo, używany był również system Redmine w celu kontroli przydziału zadań i podziału pracy. W trakcie roku okazało się jednak, iż ilość czasu, którą trzeba włożyć w zarządzanie projektem za pomocą tego systemu jest nieproporcjonalnie duża w stosunku do wielkości zespołu.

Podsumowując, tworzenie serwisu AI-Arena podczas zajęć ZPP było bardzo cennym doświadczeniem. Podczas pracy rozwinęli oni nie tylko umiejętności programistyczne, ale przede wszystkim umiejętność pracy w zespole, podziału obowiązków oraz samodzielnego prowadzenia projektu, łącznie z jego wdrożeniem. Wierzymy, że zdobyta wiedza będzie pomocna w przyszłym życiu zawodowym, zarówno w pracy jak i w zarządzaniu własnymi projektami.

Rozdział 14

Spis płyty

Dodatek A

Przykładowe gry

A.1. Szachy

Znana na całym świecie gra dwuosobowa, rozgrywana na kwadratowej planszy o 64 polach. Gracze wykonują naprzemienne posunięcia figurami, a gra kończy się remisem, bądź zwycięstwem któregoś z graczy. Gra jest całkowicie deterministyczna, ale jej złożoność nie pozwoliła dotychczas na znalezienie uniwersalnej strategii wygrywającej, bądź nieprzegrywającej. Jednak obecne programy są już od wielu lat w stanie wygrać z najlepszymi graczami ludzkimi.

A.2. Poker

Gra wieloosobowa, zawierająca elementy hazardu. Rozgrywka zaczyna się od losowego przydziału kart graczom, dlatego nie istnieje jedyna optymalna strategia grania. Celem gry jest zdobycie wszystkich punktów posiadanych przez innych graczy, które są potrzebne, by uczestniczyć w kolejnych rozdaniach. Istnieje wiele wariantów Pokera, najbardziej znanym jest tzw Texas Hold'em.

A.3. Gra giełdowa

Każdy gracz ma początkowo określone fundusze. Następnie inwestuje on w akcje dostępne na giełdzie. W zależności od inwestycji innych graczy, zmienia się wartość akcji, a co za tym idzie, dorobek każdego z graczy. Celem jest oczywiście powiększenie swojego kapitału o jak największą sumę, w kolejnych rundach.

A.4. Gra w wybory

Programy sterują fikcyjnymi partiami politycznymi. Mają ograniczony budżet, który mogą przeznaczać na reklamy, agencje public relations, preparowanie zarzutów pod adresem przeciwniej partii itp. Jest przyjęty konkretny model tego, jak akcje wpływają na wyniki wyborów, ale programy go nie znają. Mogą analizować efekty swoich działań na podstawie sondaży, które pojawiają się po kolejnych rundach. Wygrywa program, który uzyska największe poparcie w wyborach.

Dodatek B

Przykładowe programy

Dodatek C

Przebieg przykładowego turnieju

Bibliografia