

Pokemony 7

Tematyka

Pokemony to stworzenia, które trenerzy pokemonów łapią, z którymi podróżują, pracują, bawią się i za pomocą których rozgrywają walki z innymi trenerami. Trenerzy do tego stopnia lubią pojedynki, że powstało zapotrzebowanie na grę online symulującą walki pokemonów.

Jako pracownik Silph Company otrzymujesz zlecenie stworzenia prototypu aplikacji do symulacji walk pokemonów. Aplikacja ta składa się z kilku części, od zarządzania użytkownikami, po sam symulator walk w czasie rzeczywistym. Niedokończoną wersję tej aplikacji WWW z wstępnie wypełnioną bazą danych znajdziesz pod adresem



<http://www.mimuw.edu.pl/~ciebie/wwwegz2018L-start.zip>.

Dane

Otrzymujesz niedokończoną aplikację WWW napisaną w Pythonie 3, Django 2.06 i Django Channels 2.1.2. Zawiera ona wypełnioną bazę danych z poniższymi modelami.

Model Trainer

user (foreign key, primary key)	name	surname	wins	defeats	pokemon (foreign key)
2	Ash	Ketchum	42	32	8
3	Gary	Oak	40	31	5
4	Samuel	Oak	11	21	9
...	

Model ten zawiera statystyki wygranych (`wins`) i przegranych (`defeats`) walk w grze oraz posiadanego pokemona (`pokemon`). Pole `user` wskazuje na odpowiedni obiekt z modelu `django.contrib.auth.models.User`. Nie każdy użytkownik ma rekord w tabeli Trainer (np. administratorzy).

Model Pokemon

id (primary key)	name (unique)	health	attacks (many-to-many)
5	Eevee	55	2, 6, 12

8	Pikachu	35	2, 6, 14, 15
9	Psyduck	50	6, 10, 11
...	

Model Attack

id (primary key)	name (unique)	strength
2	Tackle	7
6	Body Slam	17
14	Thunderbolt	19
...

Każdy pokemon posiada 1-4 ataki.

Początkowo aplikacja WWW posiada wypełnioną bazę danych, niefunkcjonalną stronę główną, niedokończoną aplikację RESTową i niedokończoną aplikację WebSocketową.

Zadanie na 3

Na stronie głównej umieść nie ponumerowaną listę (odpowiednim znacznikiem HTML) z imionami, nazwiskami i nazwami użytkowników trenerów tworzącymi napis w postaci:

Imię Nazwisko <Nazwa użytkownika>

Na przykład:

- Ash Ketchum <ash42>
- Gary Oak <gary42>

Ustaw wysokość wiersza w liście trenerów na stronie głównej na dwukrotność rozmiaru czcionki, gdy szerokość ekranu jest ostro mniejsza od 800 pikseli. W przeciwnym razie ustaw ją na 1.5-krotność rozmiaru czcionki. Umieść na stronie głównej stopkę napisaną kursywą zawierającą Twoje *imię i nazwisko*.

Nazwy użytkowników na tej liście powinny być linkiem do podstrony każdego trenera. Każdy trener powinien posiadać swoją podstronę, na której będą wypisane jego dane: imię, nazwisko, nazwa użytkownika, procentowa ilość wygranych walk (zaokrąglona w dół, 100% w przypadku braku rozegranych walk) oraz nazwa posiadanego przez niego pokemona.

Zawrzyj treści na stronie głównej i podstronach trenerów w odpowiednich semantycznych tagach HTML. Użyj na stronie głównej co najmniej tagów `header`, `nav`, `section` i `footer`. Zadbaj, by każda strona była napisana w poprawnym HTML 5 i w poprawnym CSS. Nie wyłączaj zabezpieczenia przed atakami CSRF przy korzystaniu z mechanizmu sesji na serwerze.

Dodatki na wyższą ocenę

$ocena = \max(2, \min(5, egz_zad - egz_błędy) + lab_zad4 * 0.5)$, $egz_zad \in [2, 5.5]$

(+0.5) Umieść formularz typu POST do rejestracji nowych trenerów, tj. tworzący rekordy zarówno do modelu `User` jak i `Trainer`. Formularz powinien zawierać nazwę użytkownika, imię, nazwisko i e-mail. Nie dopuść do sytuacji, w której tylko jeden z rekordów został zapisany. Każdy nowo zarejestrowany użytkownik powinien mieć przydzielonego pokemona o nazwie `Bulbasaur`.

Podpowiedź: może się przydać [UserCreationForm](#).

(+0.5) Zrób tak, by po kliknięciu na nazwę pokemona na podstronie trenera zostało wysłane zapytanie AJAX na serwer, które ściągnie jego statystykę `health` oraz listę nazw i sił ataków klikniętego pokemona w formacie JSONowym. Wypisz uzyskane dane pod nazwą pokemona. W przypadku błędu (np. braku połączenia z serwerem) wypisz komunikat "Error".

(+0.5) Dokończ implementację statycznej strony do zdobywania nowych pokemonów znajdującą się pod adresem `/static/lottery.html`. Znajdziesz tam formularz z polem tekstowym na nazwę użytkownika, hasło oraz guzikiem do losowania pokemona. Przy wysłaniu poprawnych danych powinno zostać wysłane na serwer zapytanie, które z prawdopodobieństwem 20% zastąpi obecnego pokemona trenera losowym pokemonem wybranym z rozkładem jednostajnym z (zawsze niepustej) tabeli `Pokemon` oraz poinformuje klienta o nazwie wylosowanego pokemona lub porażce losowania. W przypadku błędu (np. niepoprawnego hasła, ale nie w przypadku porażki w losowaniu) wypisz komunikat "Error". Zapewnij bezstanowość po stronie serwera, m.in. nie korzystając po stronie serwera z mechanizmu sesji ani z obiektu `request.user`. Na stronie znajdziesz również informację z statystykami wyników losowań. Zbieraj po stronie klienta statystyki poprawnie wysłanych żądań i sukcesów w losowaniu, przechowuj je w `localStorage` i aktualizuj je po każdym poprawnym losowaniu (inicjalizując je na 0 za pierwszym razem).

(+1.0) Dokończ implementację strony do symulacji pojedynków znajdujące się pod adresem `/fight/` korzystając z WebSocketów. Pojedynek odgrywa się w następujący sposób:

1. Dwaj gracze wpisują ten sam (arbitralnie przez nich wybrany) 8-literowy kod gry i klikają na guzik "Start!".

Uruchomiona wtedy zostanie funkcja JavaScriptowa `start(code)`. Zaimplementuj ją.

2. W momencie gdy serwer połączy się z dwoma graczami, rozpoczyna się pojedynek pokemonów obu graczy z ilością punktów zdrowia równą statystyce `health` ich pokemonów.

Aby rozpocząć pojedynek wywołaj funkcję `setup(myPokemonName, myPokemonHealth, enemyPokemonName, enemyPokemonHealth, attacks)`, gdzie `myPokemonName` i `enemyPokemonName` to pola `name` używanych przez gracza i przeciwnika pokemonów, `myPokemonHealth` i `enemyPokemonHealth` to pola `health` tych pokemonów, a `attacks` to tablica stringów z nazwami ataków pokemona gracza.

3. W każdej turze gracze otrzymują do wyboru listę ataków ich pokemonów, klikają na jeden z nich i czekają aż drugi gracz to zrobi.
setup albo performAttack umożliwią graczowi wybranie ataku. Gdy gracz kliknie na atak, zostanie wywołana funkcja selectAttack(attack) (gdzie attack to string z wybraną nazwą ataku), a guziki zostaną zablokowane. Zaimplementuj selectAttack.
4. Gracze atakują się nawzajem zmniejszając ilość punktów zdrowia przeciwnika o wartość strength wybranego ataku.
Aby wykonać atak po stronie klienta, wywołaj funkcję performAttack(myPower, enemyPower), która w parametrze myPower otrzyma siłę ataku gracza, a w parametrze enemyPower otrzyma siłę ataku przeciwnika, oba uzyskane z serwera w selectAttack.
5. Jeżeli liczba punktów obu graczy jest dodatnia, następuje powrót do punktu 3.
6. Jeżeli obaj gracze uzyskają niedodatnią ilość punktów życia, następuje remis. W przeciwnym razie wygrywa gracz z dodatnią ilością punktów życia, a przegrywa jego przeciwnik. Informacja o wygranej/przegranej/remisie wyświetla się każdemu uczestnikowi gry oraz dolicza się do pola wins/defeats każdego gracza.
Wynik gry po stronie klienta zostanie automatycznie obliczony i wyświetlony w performAttack. Twoim zadaniem jest aktualizacja tych danych na serwerze.

Gotowe już jest logowanie się użytkowników, interfejs graficzny gry, logika gry po stronie klienta i szablon klasy do obsługi WebSocketów po stronie serwera (FightConsumer). Twoim zadaniem jest modyfikacja kodu JavaScript tak, by obsługiwał brakującą wymianę informacji oraz uzupełnienie całego kodu obsługi WebSocketów na serwerze tak, by przekazywał informacje użytkownikom, komunikował się z obiektem obsługującym drugiego klienta, pobierał z bazy danych odpowiednie dane, liczył stan gry i aktualizował statystyki po zakończeniu gry.

Skonfiguruj Redisa do używania domyślnego serwera na localhost:6379 i wykorzystaj mechanizmy z Django Channels do komunikacji między wątkami. Do synchronizacji i komunikacji między wątkami używaj wyłącznie Django Channels. Możesz założyć, że każda wiadomość wysłana przez Redisa dojdzie do odbiorcy. Nie musisz informować o zainstniących błędach żadnego z podłączonych klientów. Możesz założyć, że nie jest możliwe, by więcej niż dwóch graczy wpisało ten sam kod gry. Nie aktualizuj informacji o wygranej na serwerze w oparciu o dane klienta, bo mogą być one sfalszowane.

W klasie FightConsumer jest trochę wykomentowanego kodu, który może Ci pomóc w rozwiązaniu zadania.

Reguły

Niedozwolone są jakiekolwiek formy komunikacji. Można pracować na własnym sprzęcie i korzystać z swoich danych na studentsie, swoim laptopie, lub w postaci papierowej.

Można korzystać z następujących stron internetowych:

- <https://www.djangoproject.com/>
- <http://jquery.com/>
- <http://getbootstrap.com/>
- <http://www.w3schools.com/>, <http://www.w3.org/>
- https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
- <https://virtualenv.pypa.io/en/latest/>

- <https://docs.python.org/3/>
- <https://stackoverflow.com/> (bez postowania!)
- <https://channels.readthedocs.io/en/latest/>
- <https://redis.io/>, <https://github.com/MicrosoftArchive/redis>, lub inny port
- <http://bulbapedia.bulbagarden.net/>
- Oficjalne dokumentacje używanych bibliotek i narzędzi
- <https://www.google.pl/> (w sensownym zakresie, jeśli ktoś nie wie, co to jest sensowny zakres, to dla niego sensowny zakres ogranicza się do poprzednich stron)

W szczególności nie jest dozwolone korzystanie z tutoriali innych niż znajdujące się powyżej. Oficjalne dokumentacje użytych bibliotek są ok.

Można serwować jQuery bądź Bootstrap z zewnętrznych serwerów. Do rozwiązania warto dodać README.txt opisujący co się zrobiło i dlaczego. Oprócz funkcjonalności będzie oceniany dobór metody i jakoś realizacji.

Oddawanie

```
zip -r egz_www_2018.zip katalog_z_egzaminem
```

Jeśli egz_www_2018.zip ma więcej niż 3MB, to coś za dużo zzipowałeś. Umieść spakowany plik tak, aby był dostępny pod adresem:

```
http://students.mimuw.edu.pl/~twoj_login/egz_www_2018.zip
```

```
cp egz_www_2018.zip ~/public_html/
chmod a+r ~/public_html/egz_www_2018.zip
chmod a+x ~ ~/public_html
```

Jeżeli używasz bibliotek Pythonowych innych niż django, channels i channels-redis, załącz o tym informację aktualizując requirements.txt np. w następujący sposób:

```
pip freeze > katalog_z_egzaminem/requirements.txt
```

O godzinie 12:30 skrypt pobierze Twoje rozwiązanie z katalogu domowego i wyśle Ci maila. Dopiero po odebraniu maila wyjdź z sali.