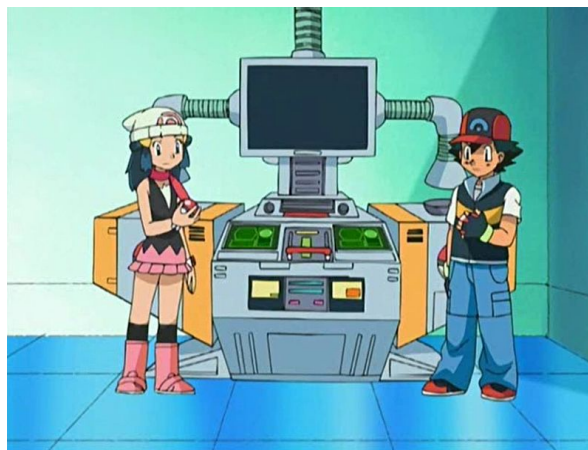


Pokemony 5

Tematyka

Pokemony to stworzenia, które trenerzy pokemonów łapią, z którymi podróżują, pracują, bawią się i za pomocą których rozgrywają walki z innymi trenerami. Od czasu do czasu chcą wymienić się swoimi pokemonami z innymi trenerami za pomocą specjalnych maszyn. Pokemony potrafią również ewoluować, czyli zmieniać się w swoje silniejsze wersje. Jest kilka pokemonów, które ewoluują w momencie, gdy są wymieniane pomiędzy trenerami.



Twoim zadaniem jest dokończenie prototypu aplikacji WWW do rejestracji pokemonów posiadanych przez trenerów, współpracującej za pomocą RESTowego API z maszynami do wymiany pokemonów. Niedokończoną wersję aplikacji WWW znajdziesz pod adresem <http://www.mimuw.edu.pl/~ciebie/wwwegz2016L-start.zip>.

Dane

Otrzymujesz niedokończoną aplikację WWW napisaną w Pythonie 3 i Django 1.11. Zawiera ona wypełnioną bazę danych z następującymi modelami:

Model Trainer

id (primary key)	name
1	Ash Ketchum
2	Gary Oak
3	Samuel Oak
...	...

Imiona mogą się powtarzać.

Model PokemonType

id (primary key)	name (unique)	tradeEvolution (foreign key)
92	Gastly	-
93	Haunter	94
94	Gengar	-
...

Model Pokemon

id (primary key)	type (foreign key)	trainer (foreign key)
1	93	2
2	133	1
...

Jeden trener może posiadać wiele pokemonów tego samego typu.

Początkowo aplikacja WWW posiada wypełnioną bazę danych, niefunkcjonalną stronę główną, niedokończoną obsługę RESTowego API i niedokończoną statyczną stronę do testowania RESTowego API.

Zadanie na 3

Wypisz na stronie głównej listę imion wszystkich trenerów pokemonów. Każde imię powinno być linkiem do podstrony tego trenera. Stwórz podstrony trenerów zawierające ich imię i listę **nazw** posiadanych przez nich pokemonów (**wszystkich, bez distinct**). Na górze strony trenerów pokemonów dodaj linka do powrotu do strony głównej. Na każdej z tych stron (**liście trenerów i podstronie trenera**) umieść wycelowaną w poziomie stopkę, na której będzie napisane (wraz ze znakami <, >):

<<< Projekt sfinansowany przez Ligę Pokemonów >>>

Zawrzyj treść na stronach w odpowiednich semantycznych tagach HTML. Użyj co najmniej tagów header, nav, section i footer (niekoniecznie na obu rodzajach stron). Użyj CSS tak, by wszystkie linki były **zielone** przed i po odwiedzeniu ich, były **podkreślone po najechaniu na nie myszką** i **nie były podkreślone w przeciwnym przypadku**.

(no dobra, zablokowali nas, wycofujemy to wymaganie) ~~Zadbaj, by każda strona walidowała się bez warningów jako poprawny HTML 5 w <https://validator.w3.org/>.~~

Dodatki na wyższą ocenę

(+0.5) Dodaj do strony każdego trenera formularz (zwykły, nie AJAX), za pomocą którego będzie można dodać nowego pokemona należącego do tego trenera do bazy danych. Formularz powinien zawierać jedno (**widoczne**) pole - listę rozwijaną (<select>) z nazwami typów pokemonów. Formularz powinien wysyłać dane metodą POST i mieć włączone domyślne zabezpieczenie przed atakami CSRF. Sprawdzaj po stronie serwera poprawność wysyłanych danych tak, aby `ForeignKey` zawierał wpisy do istniejących trenerów i pokemonów. W przypadku błędu wypisz jakiś komunikat o błędzie użytkownikowi (np. "Error").

(+0.5) Dokończ implementację RESTowego API i aplikacji testowej do wymiany pokemonów. API jest następujące: można wysłać zapytanie POST na adres `/api/trade/` z dwoma kluczami i wartościami (formularza POST), `pokemon1` i `pokemon2`, zawierającymi numery identyfikacyjne pierwszego i drugiego pokemona (`Pokemon.id`)

W wyniku zapytania pokemony o numerach identyfikacyjnych `pokemon1` i `pokemon2` powinny mieć zamienionych miejscami swoich właścicieli (tj. trener pokemona `pokemon1` powinien stać się nowym właścicielem pokemona `pokemon2` i odwrotnie). Zapytanie nie powinno się udać, jeżeli właściciele obu pokemonów to ten sam trener. Dodatkowo, jeżeli pokemony posiadają ewolucję poprzez wymianę (`Pokemon.tradeEvolution`), to ich typ powinien ulec zmianie na typ ich ewolucji.

Wynikiem zapytania powinien być słownik zakodowany w JSONie z kluczami `evolved1` i `evolved2` zawierającymi informację, czy odpowiednio `pokemon1` i `pokemon2` ewoluowały podczas wymiany (0 jeżeli nie, 1 jeżeli tak). W przypadku błędu (m. in. gdy podane pokemony nie istnieją lub mają tego samego trenera) powinna zostać zwrócona pusta wiadomość (nie wiadomość z pustym słownikiem) z kodem błędu HTTP 403 Forbidden. Dokończ też implementację statycznej strony do testowania RESTowego API znajdującej się pod adresem `/static/rest-test.html`, która wysyła to zapytanie i wyświetla wynik (wszędzie operując na numerach identyfikacyjnych, a nie nazwa). W przypadku jakiegokolwiek błędu wypisz komunikat "Error". Pozostaw bez zmian brak zabezpieczenia przed CSRF.

(+0.5) Napisz trzy Djangoowe unit testy (uruchamiane przez `./manage.py test`):

- Test sprawdzający, czy strona główna pod adresem `/` się otwiera i zwraca kod błędu HTTP 200 OK, gdy baza danych jest pusta
- Test sprawdzający, czy strona główna pod adresem `/` się otwiera i ilość trenerów, których dane są przekazane do template tej strony wynosi 2, gdy baza danych posiada dwóch trenerów
- Test sprawdzający, czy strona wybranego trenera się otwiera i ilość pokemonów, których dane są przekazane do template tej strony wynosi 2, gdy baza danych posiada tylko tego trenera i jego dwa pokemony

Podpowiedź: może przydać się `status_code` i `context`.

(+0.5) Stwórz konto administratora o nazwie użytkownika `jenny`, e-mailu

`jenny@pokemon.gov` i hasło `growlith4`. Dla każdego modelu stwórz stronę w Panelu administracyjnym, na której będzie tabelka z wszystkimi polami z bazy danych. Zadbaj, by w tabelach modeli `PokemonType` i `Pokemon` wyświetlały się nazwy pokemonów i trenerów zamiast cyfr w polach `PokemonType.tradeEvolution`, `Pokemon.type` i `Pokemon.trainer`. Dodaj do modeli `Trainer` i `PokemonType` wyszukiwarkę tekstową po polach `name`. W modelu `PokemonType` domyślnie sortuj listę rosnąco po numerze identyfikacyjnym (`PokemonType.id`).

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 wyłącznie z następujących stron internetowych:

- <https://www.djangoproject.com/>
- <http://jquery.com/>
- <http://getbootstrap.com/>
- <http://www.w3schools.com/>, <http://www.w3.org/>
- <https://virtualenv.pypa.io/en/latest/>

- <https://docs.python.org/3/>
- <https://stackoverflow.com/> (bez postowania!)
- <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_2017.zip katalog_z_egzaminem
```

Jeśli egz_www_2017.zip ma więcej niż 1MB, to coś za dużo zzipowałeś. Umieść spakowany plik tak, aby był dostępny pod adresem:

```
http://students.mimuw.edu.pl/~twoj_login/egz_www_2017.zip
```

```
cp egz_www_2017.zip ~/public_html/
chmod a+r ~/public_html/egz_www_2017.zip
chmod a+x ~ ~/public_html
```

O godzinie 14:00 skrypt pobierze Twoje rozwiązanie z katalogu domowego i wyśle Ci maila. Dopiero po odebraniu maila wyjdź z sali.