Przejdź do głównej zawartości

Projekt

Projekty należy przygotować indywidualnie.

Q&A na temat projektu: link

Co będzie oceniane?

Model konceptualny - powinien składać się z diagramu E-R, komentarza zawierającego więzy pominięte w diagramie, oraz opisu ról wraz z funkcjonalnościami (zgodnie ze specyfikacją poniżej).

Model fizyczny - powinien być plikiem sql nadającym się do czytania (i oceny) przez człowieka. Powinien zawierać definicję wszystkich niezbędnych użytkowników bazy i ich uprawnień, tabel, więzów, indeksów, kluczy, akcji referencyjnych, funkcji, perspektyw i wyzwalaczy. Nie jest niezbędne wykorzystanie wszystkich tych udogodnień, ale tam, gdzie pasują, powinny być wykorzystywane.

Dokumentacja projektu - ma się składać z pojedynczego pliku pdf zawierającego ostateczny model konceptualny oraz dokładne instrukcje, jak należy aplikację uruchomić. Dokumentacja ma dotyczyć tego, co jest zaimplementowane; w szczególności, nie można oddać samej dokumentacji, bez projektu.

Program - kod źródłowy oraz poprawność i efektywność działania.

Oddajemy archiwum zawierające wszystkie pliki źródłowe programu, dokumentację w pliku pdf, model fizyczny w pliku sql oraz polecenie typu make (ew. skrypt run.sh, itp.) umożliwiające kompilację i uruchomienie systemu.

Projekt: System wspomagający organizację konferencji

Uwaga: Pogrubioną czcionką - dodatkowe wyjaśnienia dodane po ogłoszeniu specyfikacji.

Jesteś odpowiedzialny za implementację systemu wspomagającego organizację i przeprowadzenie konferencji. Twoim zadaniem jest zaimplementowanie zdefiniowanego poniżej API.

System ma udostępniać API niezależnym aplikacjom działającym na urządzeniach mobilnych uczestników konferencji. Niemniej jednak, ze względu na to, że interesuje nas tematyka baz danych przedmiotem projektu jest stworzenie wersji deweloperskiej systemu, w której wywołania API będą wczytywane z dostarczonego pliku.

Opis problemu

Konferencja obejmuje od kilku do kilkunastu wydarzeń, z których każde zawiera do kilkudziesięciu referatów. Każde wydarzenie posiada datę rozpoczęcia i zakończenia, wydarzenia mogą się pokrywać.

Uczestnicy (kilkaset osób) rejestrują się na dowolnie wybrane przez siebie wydarzenia. Każdy uczestnik może wygłaszać dowolną liczbę referatów (być może 0). Ponadto uczestnicy mogą nawiązywać (symetryczne) znajomości z innymi uczestnikami.

Uczestnicy korzystają z systemu poprzez aplikację na urządzeniu mobilnym, która umożliwia przeglądanie danych konferencji, planu wydarzeń dla danego uczestnika, a także proponowanie i dowiadywanie się o dodatkowych referatach organizowanych spontanicznie – uczestnik proponuje referat, organizator konferencji go zatwierdza, przydziela salę oraz upublicznia. Aplikacja rejestruje obecność uczestnika na poszczególnych referatach, uczestnik może ocenić każdy z referatów (niezależnie od tego czy referat się już odbył). Każdy uczestnik posiada unikalny login i hasło.

Organizator konferencji może definiować wydarzenia i ich zawartość, przeglądać wszystkie zbierane dane, w tym również rozmaite statystyki dotyczące aktywności uczestników.

Technologie

System Linux. Język programowania dowolny – wybór wymaga zatwierdzenia przez prowadzącego pracownię. Baza danych – postgresql.

Twój program po uruchomieniu powinien przeczytać ze standardowego wejścia ciąg wywołań funkcji API, a wyniki ich działania wypisać na standardowe wyjście.

Wszystkie dane powinny być przechowywane w bazie danych, efekt działania każdej funkcji modyfikującej bazę, dla której wypisano potwierdzenie wykonania (wartość OK) powinien być utrwalony. Program może być uruchamiany wielokrotnie np. najpierw wczytanie pliku z danymi początkowymi, a następnie kolejnych testów poprawnościowych. Przy pierwszym uruchomieniu program powinien utworzyć wszystkie niezbędne elementy bazy danych (tabele, więzy, funkcje wyzwalacze) zgodnie z przygotowanym przez studenta modelem fizycznym. Baza nie będzie modyfikowana pomiędzy kolejnymi uruchomieniami. Program nie będzie miał praw do ~~**czytania,**~~ tworzenia i zapisywania jakichkolwiek plików. **Program będzie mógł czytać pliki z bieżącego katalogu.**

Format pliku wejściowego

Każda linia pliku wejściowego zawiera obiekt JSON (http://www.json.org/json-pl.html). Każdy z obiektów opisuje wywołanie jednej funkcji API wraz z argumentami.

Przykład: obiekt { "function": { "arg1": "val1", "arg2": "val2" } } oznacza wywołanie funkcji o nazwie function z argumentem arg1 przyjmującym wartość "val1" oraz arg2 – "val2".

W pierwszej linii wejścia znajduje się wywołanie funkcji open z argumentami umożliwiającymi nawiązanie połączenia z bazą danych.

Przykładowe wejście

```
{ "open": { "baza": "stud", "login": "stud", "password": "d8578edf8458ce06fbc"}}
{ "function1": { "arg1": "value1", "arg2": "value2" } }
{ "function2": { "arg1": "value4", "arg2": "value3", "arg3": "value5" } }
{ "function3": { "arg1": "value6" } }
{ "function4": { } }
```

Format wyjścia

Dla każdego wywołania wypisz **w osobnej linii** obiekt JSON zawierający status wykonania funkcji OK/ERROR/NOT IMPLEMENTED oraz zwracane dane wg specyfikacji tej funkcji.

Format zwracanych danych (dla czytelności zawiera zakazane znaki nowej linii):

```
{ "status":"OK",

  "data": [ { "atr1":"v1",

              "atr2":"v2" },

            { "atr1":"v3",

              "atr2":"v3" } ]
}
```

Tabela data zawiera wszystkie wynikowe krotki. Każda krotka zawiera wartości dla wszystkich atrybutów.

Przykładowe wyjście

```
{ "status": "OK" }
{ "status": "NOT IMPLEMENTED" }
{ "status": "OK", "data": [ { "a1": "v1", "a2": "v2"}, { "a1": "v3", "a2": "v3"} ] }
{ "status": "OK", "data": [ ] }
{ "status": "ERROR" }
```

Format opisu API

Oznaczenia:

O – wymaga autoryzacji jako organizator, U – wymaga autoryzacji jako zwykły uczestnik, N – nie wymaga autoryzacji, * - wymagana na zaliczenie

<function> <arg1> <arg2> … <argn> // nazwa funkcji oraz nazwy jej argumentów

// opis działania funkcji

// opis formatu wyniku: lista atrybutów wynikowych tabeli data

Aby uzyskać 10 punktów za część projekt (warunek zaliczenia) należy oddać samodzielnie napisany program implementujący wszystkie funkcje oznaczone znakiem *, dokumentację (zawierającą model konceptualny) oraz model fizyczny.

Nawiązywanie połączenia i definiowanie danych organizatora

```
(*) open <baza> <login> <password>
// przekazuje dane umożliwiające podłączenie Twojego programu do bazy - nazwę bazy, login oraz hasło, wywoływane dokładnie jeden raz, w pierwsz
// zwraca status OK/ERROR w zależności od tego czy udało się nawiązać połączenie z bazą

(*) organizer <secret> <newlogin> <newpassword>
// tworzy uczestnika <newlogin> z uprawnieniami organizatora i hasłem <newpassword>, argument <secret> musi być równy d8578edf8458ce06fbc5bb76a
```

Operacje modyfikujące bazę

Każde z poniższych wywołań powinno zwrócić obiekt JSON zawierający wyłącznie status wykonania: OK/ERROR/NOT IMPLEMENTED.

```
(*O) event <login> <password> <eventname> <start_timestamp> <end_timestamp> // rejestracja wydarzenia, napis <eventname> jest unikalny

(*O) user <login> <password> <newlogin> <newpassword> // rejestracja nowego uczestnika <login> i <password> służą do autoryzacji wywołującego f

(*O) talk <login> <password> <speakerlogin> <talk> <title> <start_timestamp> <room> <initial_evaluation> <eventname> // rejestracja referatu/za

(*U) register_user_for_event <login> <password> <eventname> // rejestracja uczestnika <login> na wydarzenie <eventname>

(*U) attendance <login> <password> <talk> // odnotowanie faktycznej obecności uczestnika <login> na referacie <talk>

(*U) evaluation <login> <password> <talk> <rating> // ocena referatu <talk> w skali 0-10 przez uczestnika <login>

(O) reject <login> <password> <talk> // usuwa referat spontaniczny <talk> z listy zaproponowanych,

(U) proposal  <login> <password> <talk> <title> <start_timestamp> // propozycja referatu spontanicznego, <talk> - unikalny identyfikator refera

(U) friends <login1> <password> <login2> // uczestnik <login1> chce nawiązać znajomość z uczestnikiem <login2>, znajomość uznajemy za nawiązana
```

Pozostałe operacje

Każde z poniższych wywołań powinno zwrócić obiekt JSON zawierający status wykonania OK/ERROR, a także tabelę data zawierającą krotki wartości atrybutów wg specyfikacji poniżej.

```
(*N) user_plan <login> <limit> // zwraca plan najbliższych referatów z wydarzeń, na które dany uczestnik jest zapisany (wg rejestracji na wydar
// Atrybuty zwracanych krotek:
   <login> <talk> <start_timestamp> <title> <room>

(*N) day_plan <timestamp> // zwraca listę wszystkich referatów zaplanowanych na dany dzień posortowaną rosnąco wg sal, w drugiej kolejności wg
//  <talk> <start_timestamp> <title> <room>

(*N) best_talks <start_timestamp> <end_timestamp> <limit> <all> // zwraca referaty rozpoczynające się w  danym przedziale czasowym posortowane
//  <talk> <start_timestamp> <title> <room>

(*N) most_popular_talks <start_timestamp> <end_timestamp> <limit> // zwraca referaty rozpoczynające się w podanym przedziału czasowego posortow
//  <talk> <start_timestamp> <title> <room>

(*U) attended_talks <login> <password> // zwraca dla danego uczestnika referaty, na których był obecny
//  <talk> <start_timestamp> <title> <room>

(*O) abandoned_talks <login> <password>  <limit> // zwraca listę referatów posortowaną malejąco wg liczby uczestników <number> zarejestrowanych
//  <talk> <start_timestamp> <title> <room> <number>
```

```
(N) recently_added_talks <limit> // zwraca listę ostatnio zarejestrowanych referatów, wypisuje ostatnie <limit> referatów wg daty zarejestrowan
//  <talk> <speakerlogin> <start_timestamp> <title> <room>

(U/O) rejected_talks <login> <password> // jeśli wywołujący ma uprawnienia organizatora zwraca listę wszystkich odrzuconych referatów spontanic
//  <talk> <speakerlogin> <start_timestamp> <title>

(O) proposals <login> <password> // zwraca listę propozycji referatów spontanicznych do zatwierdzenia lub odrzucenia, zatwierdzenie lub odrzuce
//  <talk> <speakerlogin> <start_timestamp> <title>

(U) friends_talks <login> <password> <start_timestamp> <end_timestamp> <limit> // lista referatów  rozpoczynających się w podanym przedziale cz
//  <talk> <speakerlogin> <start_timestamp> <title> <room>

(U) friends_events <login> <password> <eventname> // lista znajomych uczestniczących w danym wydarzeniu
//  <login> <eventname> <friendlogin>

(U) recommended_talks <login> <password> <start_timestamp> <end_timestamp> <limit> // zwraca referaty rozpoczynające się w podanym przedziale c
//  <talk> <speakerlogin> <start_timestamp> <title> <room> <score>
```

Q&A na temat projektu: link

Ostatnia modyfikacja: czwartek, 25 maj 2017, 22:47