Bazy danych

System wspomagający organizację konferencji

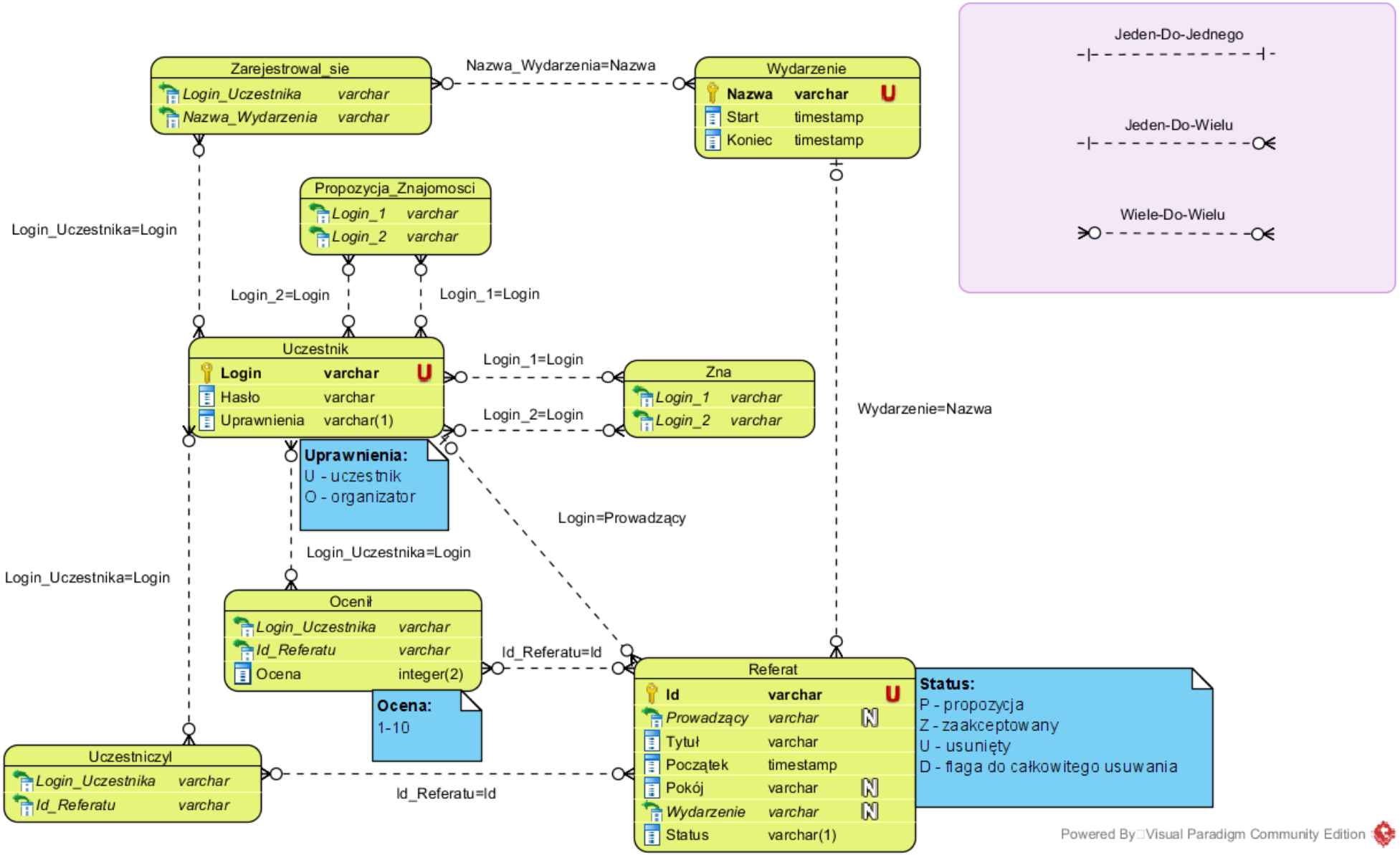
Kamil Matuszewski 272331

6 czerwca 2017

1 Model konceptualny

Obrazek przedstawiający model konceptualny można znaleźć na następnej stronie. Mamy osiem encji:

- **uczestnik(login,haslo,uprawnienia)** - Przechowuje informacje o uczestnikach a także o organizatorach (opis ról poniżej). Login jest unikalnym identyfikatorem.
- **wydarzenie(nazwa,start,koniec)** - Przechowuje informacje o wydarzeniach - nazwa jest unikalnym identyfikatorem, a start i koniec to kolejno data początku i końca wydarzenia.
- **referat(id,prowadzacy,tytul,poczatek,pokoj,wydarzenie,status)** - Przechowuje informacje o kolejnych referatach. Id jest unikalnym identyfikatorem. Referat może nie należeć do żadnego wydarzenia, ale jeśli należy, to czas rozpoczęcia musi być z przedziału [początek,koniec] odpowiadającego mu wydarzenia. Jeśli status jest równy p, to poczatek, pokoj i wydarzenie są nullami. Status d jest używany tylko do usuwania krotki, w przeciwnym wypadku przy usuwaniu zmieniamy status na "u".
- **propozycja_znajomosci(login_1,login_2)** - Przechowuje informacje o propozycjach znajomości, jeśli propozycja jest symetryczna to odpowiednie krotki są usuwane, a informacja o znajomości dodawana jest do tabeli **zna**. Oba atrybuty są kluczami obcymi odnoszącymi się do encji **uczestnik**.
- **zna(login_1,login_2)** - Informacja mówiąca, że użytkownik login_1 zna użytkownika login_2, znajomość jest symetryczna. Oba atrybuty są kluczami obcymi odnoszącymi się do encji **uczestnik**.
- **zarejestrowal_sie(login_uczestnika,nazwa_wydarzenia)** - Informacja o tym, że użytkownik zarejestrował się na dane wydarzenie. Pierwszy atrybut jest kluczem obcym wskazującym na encję **uczestnik**, drugi jest kluczem obcym wskazującym na encję **wydarzenie**.
- **uczestniczyl(login_uczestnika,id_referatu)** - Informacja o tym, że użytkownik uczestniczył w referacie (odnotowana jest faktyczna obecność). Pierwszy atrybut jest kluczem obcym wskazującym na encję **uczestnik**, drugi jest kluczem obcym wskazującym na encję **referat**.
- **ocenil(login_uczestnika,id_referatu,ocena)** - Informacja o ocenie danego referatu przez danego użytkownika. Pierwszy atrybut jest kluczem obcym wskazującym na encję **uczestnik**, drugi jest kluczem obcym wskazującym na encję **referat**, trzeci jest oceną referatu w skali 1-10.


Jeden-Do-Jednego
Jeden-Do-Wielu
Wiele-Do-Wielu
Zarejestrowal_sie
Login_Uczestnika varchar
Nazwa_Wydarzenia varchar
Nazwa_Wydarzenia=Nazwa
Wydarzenie
Nazwa varchar U
Start timestamp
Koniec timestamp
Propozycja_Znajomosci
Login_1 varchar
Login_2 varchar
Login_Uczestnika=Login
Login_2=Login
Login_1=Login
Uczestnik
Login varchar U
Hasło varchar
Uprawnienia varchar(1)
Login_1=Login
Login_2=Login
Zna
Login_1 varchar
Login_2 varchar
Wydarzenie=Nazwa
Uprawnienia:
U - uczestnik
O - organizator
Login=Prowadzący
Login_Uczestnika=Login
Login_Uczestnika=Login
Ocenił
Login_Uczestnika varchar
Id_Referatu varchar
Ocena integer(2)
Id_Referatu=Id
Ocena:
1-10
Referat
Id varchar U
Prowadzący varchar N
Tytuł varchar
Początek timestamp
Pokój varchar N
Wydarzenie varchar N
Status varchar(1)
Status:
P - propozycja
Z - zaakceptowany
U - usunięty
D - flaga do całkowitego usuwania
Uczestniczyl
Login_Uczestnika varchar
Id_Referatu varchar
Id_Referatu=Id
Powered By Visual Paradigm Community Edition

1.1 Opis ról

Mimo, że nie ma określonych ról w samej bazie, to dostęp do niektórych funkcji wymaga podania loginu i hasła. Login i hasło są następnie sprawdzane w bazie, i zwracana jest rola użytkownika i sprawdzane są uprawnienia do wykonania podanej funkcji. Wyróżniamy więc role:

- Uczestnik - osoba która w bazie ma status uczestnika ma dostęp do funkcji oznaczonych w specyfikacji przez "U". Tą rolę otrzymuje każdy nowy użytkownik stworzony za pomocą polecenia *user*.
- Organizator - osoba która w bazie ma status organizatora ma dostęp do funkcji oznaczonych w specyfikacji przez "O". Tę rolę otrzymują użytkownicy stworzeni za pomocą polecenia *organizer*. W szczególności organizator nie jest rozszerzeniem uprawnień uczestnika (nie ma dostępu do funkcji oznaczonych w specyfikacji przez "U").

2 Model fizyczny

Model fizyczny wraz z odpowiednimi komentarzami znajduje się w pliku **fizyczny.sql**. Funkcje są odpowiednio ponazywane a także posiadają komentarze mówiące, do którego z zapytań się odnoszą. Do tego utworzone zostały odpowiednie typy, by funkcje mogły zwrócić odpowiednio przemianowane krotki (dla ułatwienia działania aplikacji).

3 Aplikacja

Aplikacja okienkowa napisana jest w technologii .NET Core. Użyte biblioteki:

- *Npgsql* - biblioteka do łączenia się z bazą danych.
- *Newtonsoft.Json* - biblioteka do przetwarzania zapytań Json a także do tworzenia obiektów wynikowych.
- *Dapper* - biblioteka do wykonywania zapytań a także do zbierania wyników.

3.1 Działanie aplikacji

Podczas każdego uruchomienia, program czyta pierwszą linię i (zgodnie ze specyfikacją) traktuje ją jak Jsonowe zapytanie "open". Zapytanie to otwiera bazę danych, tworząc nowy obiekt *Database*. Obiekt ten przechowuje wszystkie potrzebne informacje, zawiera też metody umożliwiające wykonywanie poszczególnych zapytań do bazy danych. Jeśli otwarta baza nie zawiera tabeli "uczestnik", to znaczy, że jest pusta (zgodnie ze specyfikacją), wczytujemy więc model fizyczny z pliku **fizyczny.sql**.

Jeśli wszystko było w porządku, to w pętli wczytujemy i przetwarzamy kolejne linie. Poszczególne odpowiedzi wrzucamy na kolejkę, by następnie wypisać wszystkie zapytania pod koniec działania programu.

Każde zapytanie jest odpowiednio przetwarzane w celu zdobycia nazwy funkcji a także wszystkich potrzebnych danych a następnie umieszczone w obiekcie **Input**. Wywoływana jest odpowiednia funkcję i po wykonaniu zapytania do bazy danych zwracamy obiekt typu *Output* zawierający status (*OK/ERROR/NOT IMPLEMENTED*) oraz wynik zapytania jeśli jest to wymagane. Następnie z obiektu tworzony jest Json, który wrzucany jest na kolejkę.

Aplikacja odpowiada jedynie za przetwarzanie obiektów Json i komunikacji pomiędzy użytkownikiem a bazą danych. O wszystkie zapytania dbają odpowiednie funkcje opisane w modelu fizycznym. Oznacza to, że przy zmianie modelu fizycznego nie trzeba niczego zmieniać w aplikacji, dopóki odpowiednie funkcje będą miały te same nazwy, argumenty i zwracały te same wartości. Aplikacja jest zabezpieczona przed tzw. **SQL injection** o co dba biblioteka *Dapper*.

Obiekt *Data* zawiera dane wynikowe, wczytane wprost z bazy danych. Z uwagi na to, że w jednym z zapytań argument wyjściowy musi być nazwany "event" a nazwanie w ten sposób zmiennej jest niemożliwe, zmienna jest nazwana "_event" i dopiero podczas przetwarzania obiektu do Jsona jej nazwa jest odpowiednio zmieniana.

Po przetworzeniu zapytań i wypisaniu odpowiedzi aplikacja kończy działanie.

4 Włączanie aplikacji

Aplikacja do uruchomienia wymaga zainstalowanego .NET Core, a do poprawnego działania wymaga także postgresa i pustej bądź wypełnionej zgodnie z modelem fizycznym bazy danych.

4.1 Instalacja .NET Core

Instalacja .NET Core przebiega zgodnie z instrukcją opisaną na stronie `https://www.microsoft.com/net/core`. Z uwagi na dostęp do tej instrukcji a także przez nieznajomość systemu użytkownika, nie będę opisywać kolejnych kroków instalacyjnych.

4.2 Uruchomienie

Kiedy znajdujemy się w folderze głównym projektu, podczas pierwszego uruchomienia należy najpierw wpisać polecenie **dotnet restore**, żeby pozyskać wszystkie biblioteki niezbędne do uruchomienia. By uruchomić samą aplikację należy wpisać **dotnet run**.

5 Testy

W pliku *test* znajdują się wygenerowane przeze mnie testy. Generują one niedużą bazę danych, która służyła mi za obiekt testów. Linijka 1 odpowiada za połączenie się z bazą danych (należy ją odpowiednio uzupełnić), od 2 do 1013 (włącznie) są operacje modyfikujące bazę danych. Linijki 1014-1026 to przykładowe zapytania zwracające dane. W testach nie zostały ujęte wszystkie możliwości a także nie zostały sprawdzone wszystkie przypadki brzegowe - służą one jedynie do upewnienia się, że aplikacja rzeczywiście potrafi się komunikować z bazą danych.