/*==========Projekt: System wspomagający organizację konferencji==========*/
/*==========Dokumentacja do projektu. Wykonał: Igor Tryhub, 275235==========*/

Dany projekt jest napisany w języku C# na platformie .NET. W celach deweloperskich używałem środowiska IDE "Visual Studio Code" z odpowiednim pluginem do C#. W kodzie źródłowym importuję następujące biblioteki:

using Npgsql; -> do komunikowania się z BDMS PostgreSQL
using Newtonsoft.Json; -> do konwertacji objektów do/z formatu JSON
using System.Reflection; -> do wywoływania funkcji z jej nazwy za pomocą refleksji

W projekcie zostały oprogramowane wszystkie obowiązkowe operacje (t.j. te oznaczone "(*)"). Pozostałe operacje ze specyfikacji, jeżeli nie zostały zaimplementowane, jawnie zwracają JSON ze statusem "NOT IMPLEMENTED". Każda operacja, za wyjątkiem kilku (w tym "talk()"), posiada odpowiednią funkcje SQL'ową, która jest wywoływana i obrabiana na wyższym poziomie przez odpowiednio zabezpieczone blokami "try-catch" metody C#'owe. Zarówno kod SQL'owy, jak i C# zostały dość szczegółowo skomentowane. W razie potrzeby można również odkomentować linijki kodu wypisujące błędy i ich źródła.

Żeby uruchomić program z tego projektu, należy:

1) rozpakować archiwum;
2) przejść do ścieżki gdzie umieszczona jest zawartość danego projektu (t.j. polecenie "cd .../Project" w konsoli Linux'owej);
3) po uruchomieniu dbms PostgreSQL, jeżeli jesteśmy teraz w bazie pod nazwą “student”, przełaczyć do innej bazy zanim przejdziemy do kolejnego kroku, np.:

```
\connect postgres
```

4) wyczyścić starą bazę danych (o ile istnieje) z nazwą “student” i rolę „student”:

```
DROP DATABASE student;
DROP ROLE student;
```

5) utworzyć nową rolę ‘student’, bazę danych ‘student’, przełączyć się do danej bazy danych oraz załadować model fizyczny z pliku ‘db_project_physical.sql’ tworząc typy danych, tabele, widoki, funkcje i wyzwalacze niezbędne do funkcjonowania programu w sposób zgodny ze specyfikacją projektu:

```
CREATE ROLE student SUPERUSER CREATEDB CREATEROLE LOGIN REPLICATION PASSWORD 'student';
CREATE DATABASE student WITH OWNER=student TEMPLATE template0;
\connect student
\i db_project_physical.sql
```

6) podmienić zawartość pliku "input" na zawartość rzeczywistych danych wejściowych;
7) wpisać polecenie do zbudowania projektu zamieszczonego w PWD:

```
dotnet restore
```

8) wpisać polecenie do uruchomienia projektu zamieszczonego w PWD:

```
dotnet run
```

UWAGA1: zakłada się, że wpisy do bazy mogą wykonywać się poprawnie ({“status”:”OK”}) tylko jeden raz. Przy ponownym uruchomieniu “dotnet run” na już wypełnionej bazie danych wszystkie wpisy o tych samych parametrach skutkują błędem ({“status”:”ERROR”}).

UWAGA2: w modelu fizycznym zostały dodane dodatkowe CONSTRAINT’y, które autor uważał za sensowne dla modelu z rzeczywistego świata, aczkolwiek w momencie porównywania z wynikami testów wzorcowych, wyniki programu mogą nieco się różnić.