AnoFeT – Anonymous Feedback Tool

Dima Rares, Tabusca Bogdan, Popescu Dimitrie – 2 B7

Abstract. Oportunitatea de a oferi feedback asupra unui lucru (festival, persoana, loc public) postat de un anumit user.

Keywords. : feedback, categorie, eveniment, utilizator, recenzie

Cuprins :

1. Descriere proiect

Proiectul presupune un instrument Web ce permite utilizatorilor posibilitatea de a oferi feedback pentru un anumit lucru (eveniment, persoana, pub, festival, carte, etc). Un utilizator va putea folosi aplicatia web prin intermediul unui cont. Atunci când utilizatorul utilizează pentru prima oară aplicația, se va putea înregistra completând datele necesare pentru a putea fi introdus in baza de date (conform constrângerilor impuse). Ulterior, după ce și-a creat contul se va putea loga si folosi aplicația cum se cuvine. El va putea sa caute diferite lucruri după o anumita categorie, va putea sa va diferite lucruri postate pentru a lasa o recenzie, să vizualizeze itemele lui private. Pentru fiecare actiune ce va fi realizată, postare de lucru, lăsare de review, creare de item se va acționa in baza de date pentru a mentine aceste informații, si ca aplicația sa ruleze intr-un mod normal.

2. Tehnologii utilizate

Pentru gestioarea bazei de date am ales sa folosim o baza de date relationala gestionata de sistemul Oracle Server 11g. Pentru partea de back-end am ales sa folosim **Python** (versiunea 2.7), pe partea de front-end am utilizat **HTML5,CSS3 si Javascript.**

3. Arhitectura Aplicatiei

Aplicația folosește un system response factory, care analizează request-ul provenit de la pagina web, si apelează funcția specifică acelui request, după care răspuns-ul e trimis sub forma JSON, mai mult paginile javascript-urile sunt trimise prin metoda **GET**, iar request-urile efective pe care le face un utilizator sunt procesate prin **POST**. Exemple, Login system-ul cand primeste un user si o parolă generează un token pe care îl reține într-un vector, și îil trimite la user (care îl stochează local in in browser storage), de fiecare data când user-ul intră pe pagina principală, se face o verificare a validalității token-ului respectiv. Cât despre review system, utilizatorul logat, poate crea iteme care sunt de tip publice and privat, itemele publice vor fi văzute de oricine, dar numai utilizatorii logați pot da review-uri la ele. Itemele private sunt pe bază de mail, pe scurt, numai cei care au email-urile specificate la crearea obiectului pot vedea și da review-uri la acele obiecte. Un alt lucru important este că cam toate request-urile sunt făcute prin intermediul AJAX.

4. Detalii de implementare

Fiind vorba despre un site ce oferă un serviciu de feedback, fie el și anonim, este evident că un utilizator va trebui să creeze un cont pentru a putea monitoriza propriile obiecte postate, deci necesitatea unui tabel de utilizatori, de altfel prezent in majoritatea site-urilor moderne.
Analog, prin natura serviciului oferit,era nevoie de un mod de a stoca obiectele adăugate, deci o tabela Items pentru acest scop, care deține o cheie străine către tabela users, cu id-ul userului ce a creat fiecare obiect.

Tabela **SITE_USERS** va conține utilizatorii prezenți în aplicație . În primul rând, utilizatorii vor fi identificați prin câmpul ID (NUMBER) care este si cheie primară a acestei tabele. Alte câmpuri prezente în tabelă sunt NUME (VARCHAR2(30)), reprezintă numele de familie al utilizatorului, PRENUME (VARCHAR2(50)), reprezinta prenumele utilizatorului, EMAIL (VARCHAR2(100)), reprezintă email-ul utilizatorului, DATA_NASTERE (TIMESTAMP(6)) reprezintă data de naștere a utilizatorului, USERNAME (VARCHAR2(50)), câmp unic ce va reprezenta username-ul primit de catre utilizator, PASSWORD (VARCHAR2(50)) reprezintă parola contului de utilizator.

Tabela **REVIEWS** va conține recenziile utilizatorilor asupra unui item. Câmpurile prezente aici sunt : ID(NUMBER) care este cheia primară, USER_ID(NUMBER) id-ul utilizatorului care face recenzia , CONTENT(VARCHAR2(3000)) conținutul recenziei (părerea care și-o exprimă utilizatorului), DATE_LEFT(TIMESTAMP(6)) data cand a fost lasată recenzia.

Tabela **ITEMS** va conține itemle care sunt evaluate. În tabelă vor fi prezente câmpurile : ID(NUMBER) este cheia primară, USER_ID(NUMBER) id-ul utilizatorului care postează acel item, TITLE(VARCHAR2(100)) reprezintă titlul item-ului, DESCRIPTION(VARCHAR2(3000)) reprezintă descrierea item-ului scrisă de utilizator, START_DATE(TIMESTAMP(6)) reprezintă data când este postat item-ul, VIEWS(NUMBER) număr-ul de vizualizări pe care îl primește item-ul.

Acestea sunt tabelele de bază folosite în cadrul dezvoltării aplicației, folosind ca baze de date Oracle, un model relațional a fost necesară asigurarea din start a respectării anumitor relații între tabele prin chei primare, chei străine si a altor costrîngeri, de exemplu în momentul in care un utilizator creeaza item, trebuie neapărat ca acest item sa fie menținut în tabela items dar și în tabela users, pentru a respecta relațiile dintre aceste 2 tabele.

Pe partea de back-end, s-a realizat un **DBController** pentru a realiza conexiunea dintre website si baza de date Oracle. Acesta are rolul de a realiza conexiunea, de a oferi răspunsuri la interogări, etc. Pe partea de ServerFunctions, s-au implementat fiecare funcțiile necesare pe partea de back-end. De exemplu, functiile Register.py,Items.py,Account.py fiecare avânduși rolul specific, atunci când un utilizator se înregistrează trebuie să se face mutarea si în baza de date, respectate constrângerile impuse, iar pe partea de Items la fel, fiecare postare de item, fiecare creare trebuie menținută pentru a face ca site-ul să fie cât mai funcțional.

5. Diagrame

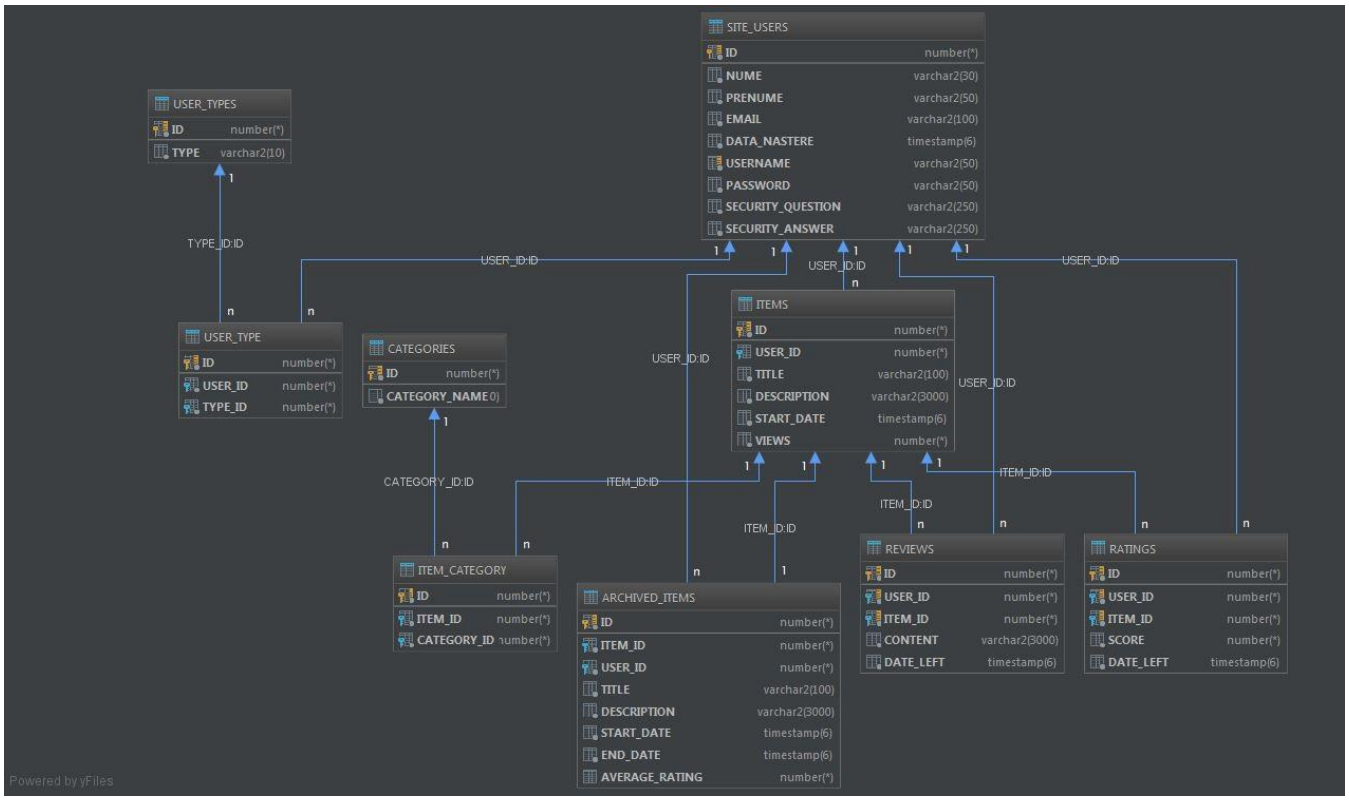

Diagrama sugerează modul de structurare a bazei de date, cum a fost gândită structurarea acesteia.

6. Concluzie

În concluzie, implementarea back-end a fost realizată in Python (versiunea 2.7) iar principalele unelte de care ne-am folosit au fost un API pentru acces-ul la baza de date Oracle, API care invelește API-ul oficial Oracle, și il face mai ușor de folosit. Iar pentru interacțiunea cu client-ul, cu front-end-ul, am folosit un API propriu care este un pachet cu diverite funcții pentru diverse funcții ale site-ului. În pachet există diverse funcții pentru login, register, items, etc necesare pe partea de front-end, care sunt optimizate deoarece nu se alocă spațiu inutiul, și menținerea structurilor repetitive la un nivel minim, acesta ducând si la un cod care este ușor de înțeles.