Bazy danych: Forum

16 stycznia 2017

Dokumentacja

Aleksandra Bielak

Spis treści

1 Projekt koncepcji, założenia

1.1 Temat projektu

Celem projektu jest stworzenie gry forumowej typu PBF (*play by forum*). Posiada ona strukturę forum, to znaczy dzieli się na podfora i tematy z postami zamieszczanymi przez różnych użytkowników. Użytkownicy posiadają profil podstawowy (login, email) oraz profil rozszerzony (karta postaci), uzupełniany podczas rejestracji wartościami wprowadzonymi przez użytkownika oraz generowanymi w bazie. Mogą także wysyłać do siebie nawzajem wiadomości. Administrator ma możliwość dodawania użytkowników na czarną listę, blokując im dostęp do gry.

1.1.1 O PBF

Na forum PBF użytkownicy wcielają się w postacie, których dane określa rozszerzony profil (tzw. karta postaci). Jako dana postać piszą posty w formie pierwszej lub trzeciej osoby oraz wdają się w interakcje z innymi graczami. Za pomocą prywatnych wiadomości mogą ustalać między sobą szczegóły gry, nieraz tworząc skomplikowane historie. Bywały przypadki, że posty graczy PBF wydawano w formie książek. Same systemy i logika gier różnią się między sobą od prostych gier tekstowych o skomplikowanej mechanice rzutu kośćmi i automatycznym przydzielaniu statystyk po typowe fora utrzymane jedynie w klimacie RPG, gdzie rozwój postaci i wynik starcia zależą od kunsztu literackiego gracza.
Tworzona w ramach projektu aplikacja należy do pierwszego z powyższych dwóch rodzajów. Zrezygnowano w niej z Mistrza Gry, a walka jest całkowicie zautomatyzowana. Użytkownik sam podemuje decyzję o zaatakowaniu innego gracza, który wypowiedział się w tym samym temacie. Wówczas na podstawie statystyk obu graczy obliczany jest wynik starcia oraz jego konsekwencje (utrata punktów życia).

1.2 Analiza wymagań użytkownika

Użytkownicy dzielą się na pięć grup: **niezalogowanych**, **zbanowanych graczy** (na czarnej liście), **graczy**, **moderatorów** i **administratorów**. Każda grupa posiada większe uprawnienia od poprzedniej Niezalogowany ma prawo:

- zarejestrować się do gry i stworzyć swój profil,
- przeglądać zawartość forów (oprócz tych z dostępem wyłącznie dla zalogowanych użytkowników),
- zalogować się (jeśli posiada konto).

Zbanowany gracz ma prawa niezalogowanego oraz może:

- pisać prywatne wiadomości do innych graczy,
- przeglądać swój profil użytkownika.

Gracz ma prawa zbanowanego gracza oraz może:

- pisać posty oraz atakować innych graczy,
- usuwać swoje posty.

Moderator ma prawa gracza oraz może:

- usuwać posty innych graczy,
- dodawać graczy na czarną listę oraz usuwać ich z niej.

Administrator ma prawa moderatora oraz może:

- usuwać tematy,
- usuwać użytkowników.

1.3 Funkcje

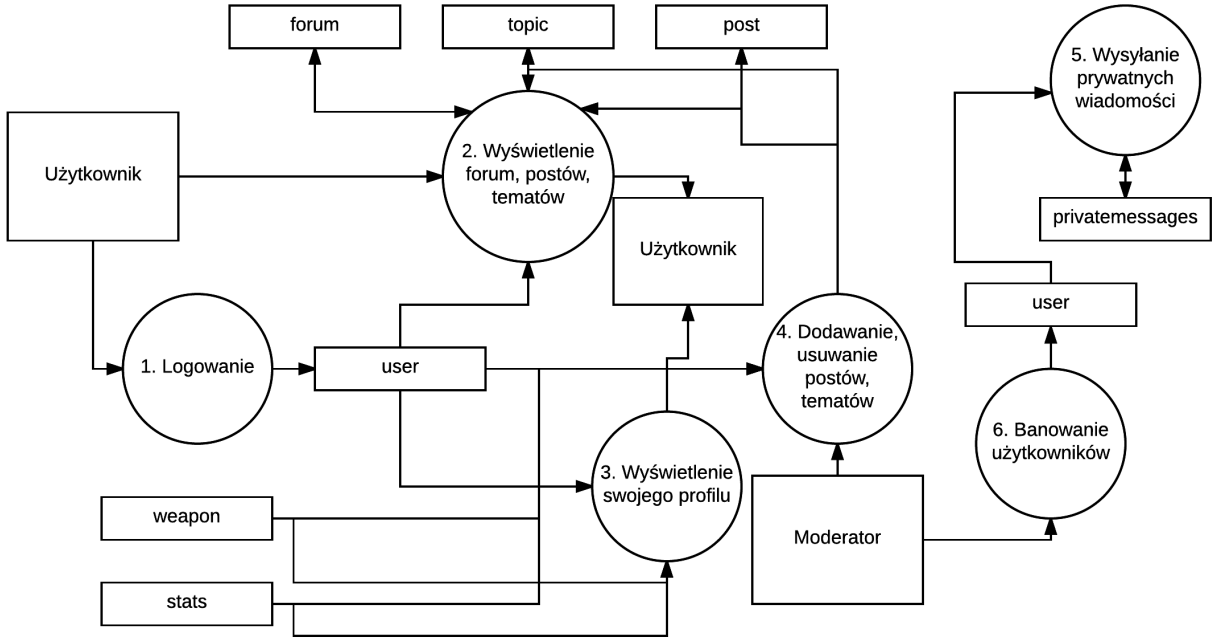
W bazie danych realizowane są realizowane funkcje związane z:

- wyświetaniem zawartości bazy,
- rejestracją i logowaniem użytkownika,
- dodawaniem nowych postów i tematów,
- rozgrywaniem walk pomiedzy graczami,
- obsługą prywatnych wiadomości.

2 Projekt diagramów

2.1 Diagram przepływu danych

Poniżej przedstawiono wygenerowany diagram DFD przedstawiający początkowe założenia projektu.



Dane może wprowadzać użytkownik lub moderator/administrator (użytkownik z dodatkowymi przywilejami). Użytkownik widzi wyświetlane dane. Na diagramie przedstawione zostały najważniejsze funkcjonalności aplikacji oraz magazyny danych, których wymagają (późniejsze encje).

2.2 Encje i ich atrybuty

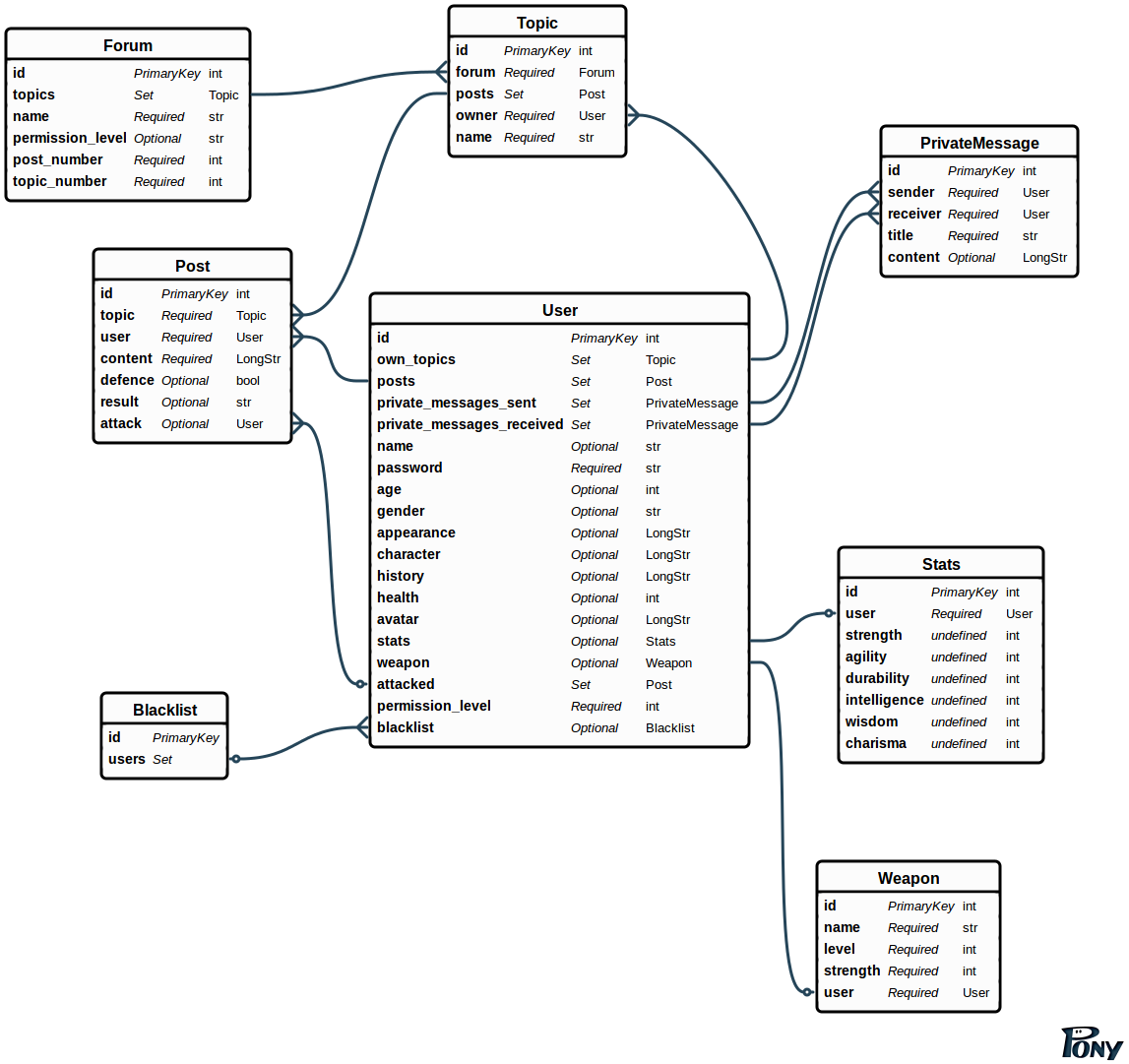
W projekcie bazy danych zdefiniowano 8 encji:

1. **Forum** - zawiera nazwę oraz ilość postów i tematów
2. **Topic** - (pl. Temat) zawiera nazwę oraz właściciela (zazwyczaj osobę, która założyła pierwszy post)
3. **Post** - posiada treść oraz dane potrzebne do walki między użytkownikami
4. **User** - posiada dane niezbędne do logowania oraz pozostałe potrzebne do gry (np. punkty zdrowia)

5. **Weapon** - (pl. Broń) - zawiera dane o broni gracza
6. **Stats** - zawiera statystyki (siłę, zwinność, wytrzymałość, inteligencję, mądrość, charyzmę)
7. **PrivateMessage** - zawiera prywatną wiadomość, jej treść, temat oraz nadawcę i odbiorcę
8. **Blacklist** - mówi, którzy użytkownicy są zbanowani (mają ograniczony dostęp do niektórych funkcji)

2.3 Relacje

Określono szczegóły encji oraz powiązania między nimi. Na ich podstawie stworzono klucze obce. Ostateczny efekt przedstawiono na poniższym diagramie ERD stworzonym w aplikacji *PonyOrm*:



3 Projekt logiczny

3.1 Projektowanie tabel, kluczy, indeksów

Na podstawie diagramu ERD wygenerowano kod SQL (*Załącznik 1*). Budowę encji i kluczy określono następującymi regułami:

- Dla każdej tabeli klucz prywatny nosi nazwę *id* i jest typu *integer*. Dzięki temu baza zachowuje jednolitą strukturę.
- Dla każdej tabeli atrybuty o podobnym znaczeniu logicznym noszą te same nazwy (np. nazwa tematu i nazwa użytkownika *name*, treść posta, prywatnej wiadomości *content*).
- Klucze obce noszą nazwę tabeli, do której się odwołują i są zawsze typu *integer*.
- Mimo iż na powyższym diagramie figuruje jednostka *Set*, wygenerowany dla bazy Postgres kod SQL zawiera już jedynie klucze obce.

W projekcie nie wykorzystano słowników danych, gdyż nie przewiduje się encji rozszerzalnych przez użytkownika (np. stała ilość statystyk, każdy użytkownik posiada jedną broń).

3.2 Analiza zależności funkcyjnych

3.2.1 Normalizacja do 3NF

Przeanalizowano po kolei wszystkie tabele badając, czy spełniają one niezbędne założenia, aby być znormalizowane do 3NF:

1. Forum - wszystkie atrybuty wtórne są zależne od klucza głównego - ilość postów i tematów w danym forum jest wyliczana na podstawie danych charakterystycznych dla TEJ jednostki.
2. Topic - tutaj sprawa się komplikuje. Mianowicie tabela zawiera zbędną daną - pole *Owner*, zatem obecnie nie można uznać tej encji za znormalizowaną. Szczegóły w analizie anomalii.
3. Post - wszystkie dane zawarte w pojedynczym poście są dla niego charakterystyczne.
4. User - dane zbędne (dotyczące broni, czy statystyk, mogące funkcjonować jako autonomiczne jednostki) przeniesiono do osobnych tabel połączonych główną relacją 1:1. Jednak nie jest to tabela znormalizowana - zawiera zbędne informacje. Więcej w następnej sekcji.
5. Weapon - zawiera jedynie dane konkretnej broni przynależnej do określonego kluczem obcym użytkownika. Jest w 3NF.
6. Stats - także jest w 3NF (analogicznie jak przy broni).
7. PrivateMessage - zawiera dane dotyczące tylko wiadomości, powiązani z nią użytkownicy są oznaczeni kluczem obcym - jest znormalizowana.
8. Blacklist - zawiera jedynie klucz obcy. W zależności od opcji wybranych w tabeli *user* może być zbędna.

3.2.2 Analiza anomalii

W trakcie analizy normalizacji wykryto w bazie następujące anomalie:

- redundancja w tabeli *topic* - w tej tabeli istnieje kolumna *owner*, która oznacza gracza będącego *właścicielem* tematu. Obecnie jest to ten sam gracz, który napisze pierwszy post, przez co informacja o właścicielu staje się zbędna, a samo bycie nim nie wiąże się z żadnymi przywilejami.
  Kolumna *owner* została stworzona na potrzeby przyszłego rozwoju aplikacji, a mianowicie, by możliwe stało się wprowadzanie tzw. krótkich sesji. Wtedy właściciel tematu zyskiwałby prawo do moderacji postów w założonym przez siebie temacie jako tzw. Mistrz Gry (obecnie funkcję tę pełni moderator i jest ona globalna).

- zbędna wartość występuje także w tabeli *User*. Obecnie pozbawienie użytkownika uprawnień odnotowywane jest w dwóch polach: *permission_level* odpowiadającym za ogólne uprawnienia (im wyższy poziom, tym wyższa ranga, od 1-4 dla zalogowanych użytkowników zgodnie z analizą wymagań użytkownika) oraz w polu *blacklist* zwierającym id czarnej listy, na której jest użytkownik (w przeciwnym wypadku jest to NULL). Obie te wartości przekazują tę samą informację - użytkownik jest zbanowany. Podobnie jak poprzednio przy tworzeniu czarnej listy wzięto pod uwagę możliwy kierunek rozwoju projektu. W planach jest wprowadzenie możliwości blokowania uprawnień na konkretnych forach, za co będą odpowiadać poszczególne czarne listy.

3.3 Operacje na danych

Logikę gry obsługuje baza danych głównie za pomocą procedur składowanych i triggerów. Poniżej przedstawiono funkcje (widoki, procedury składowane) realizujące poszczególne zadania wymienione w punkcie 1.3. Ich szczegółowy kod znajduje się w załączniku *functions.sql*

3.3.1 Wyświetlanie danych

- *displayforum* - (pl. wyświetl forum) widok, który pobiera z bazy i wysyła w odpowiednim formacie listę wszystkich forów,

- *displaytopics* - (pl. wyświetl tematy) pobiera i wysyła tematy w konkretnym forum (danym jako argument),

- *selectposts* - (pl. wybierz posty) wybiera wszystkie posty w danym temacie,

- *displayuser, displayuserposts, displayuserstats, displayuserstats, displayuserweapon* - (pl. kolejno: wyświetl użytkownika, wyświetl posty użytkownika, wyświetl tematy użytkownika, wyświetl statystykę, wyświetl broń) funkcje związane z pokazywaniem pełnego profilu użytkownika,

- *showsended, showreceived* - (pl. pokaż wysłane, pokaż odebrane) procedury służące do przeglądania historii wiadomości.

3.3.2 Rejestracja i logowanie

Dane, czy użytkownik jest zalogowany znajdują się w tablicy _SESSION w przeglądarce, ale sam proces logowania przeprowadzany jest w bazie danych i od jego wyniku zależy autoryzacja.

- *login* - służy do sprawdzenia, czy użytkownik o danym loginie i haśle istnieje w bazie, jeśli tak wydaje autoryzację, jeśli nie, odrzuca,

- *register* - przechwytuje dane wysyłane z formularza rejestracji i wprowadza użytkownika do bazy,

- *selectperms* - (pl. wybierz uprawnienia) wykonuje się zaraz po zalogowaniu. Przekazuje do tablicy _SESSION numer uprawnień użytkownika.

3.3.3 Wysyłanie wiadomości

Służy do tego funkcja sendmessage (pl. wyślij wiadomość). Pobiera ona dane z formularza i wprowadza je do odpowiedniej tabeli.

3.3.4 Dodawanie nowych postów i tematów

Liczniki postów i tematów w tabeli *forum* działają w różny sposób:

- przy dodawaniu posta (służy do tego funkcja *addpost*) wywoływany jest trigger *change_post_number*, który odpowiada za zwiększenie wartości w liczniku,
- przy dodawaniu tematu w funkcji *addtopic* znajduje kwerenda zliczająca ilość postów w temacie i ją nadpisująca.

Oprócz wspomnianych funkcji dodających (które właściwie przechwytują dane i wrzucają je do bazy) warto wspomnieć o funkcji *deletepost*, usuwającej z bazy post o podanym id. Jeśli jako rezultat akcji w danym poście gracz utracił punkty życia, nie odzyska ich z powrotem. Zapobiega to sytuacji, gdy użytkownicy wstawiają ponownie post z tą samą akcją, by uzyskać lepszy rzut kośćmi (i lepszy wynik).

3.3.5 Walka

Dodając post, gracz może zadecydować, czy chce zaatakować jakiegoś gracza (który wypowiedział się wcześniej w tym temacie) oraz czy stawia obronę. Jest to nazwane **akcją**. Podczas wstawiania posta do bazy wywoływana jest funkcja *result*, która oblicza tzw. rezultat akcji.
Rezultat zależy od rzutu sześcienną kością, statystyk, poziomu i siły broni obu użytkowników. Na tej podstawie wylicza się **moc użytkownika**. Wygrywa użytkownik o większej mocy, a przegrany traci punkty życia w ilości dziesięciokrotnie mniejszej niż różnica mocy między nim a zwycięzcą.
Jeśli atakujący postanowił się bronić, wówczas posiada 50% szans na ocalenie w wyniku przegranej. Jeśli się obroni, straci 10 razy mniej życia niż w przypadku braku ochrony, ale nie otrzyma punktów zwiększających siłę broni.

3.3.6 Funkcje dostępne jedynie dla moderatorów i administratorów

Są to procedury, które pozwalają na egzekwowanie uprawnień określonych w zbiorze wymagań (punkt 1.2):

- *deletetopic* - (pl. usuń temat) funkcja pozwalająca usunąć temat razem z wszystkimi postami w nim. Po wywołaniu tej funkcji uruchamiany jest trigger *change_post_number_when_delete*, który zmniejsza licznik postów,
- *deleteuser* - (pl. usuń użytkownika) usuwa użytkownika razem z jego bronią, statystykami, postami i prywatnymi wiadomościami,
- *banuser* - przenosi użytkownika na czarną listę i ogranicza jego uprawnienia.

4 Projekt funkcjonalny

4.1 Interfejsy obsługi danych

Użytkownik może wprowadzać dane do bazy za pomocą następujących formularzy:

- formularz logowania,
- formularz rejestracji,

- formularz tworzenia nowej wiadomości,
- formularz tworzenia nowego tematu,
- formularz tworzenia nowego postu.

O ile w przypadku dwóch pierwszych formularzy zależność jest raczej oczywista, to tworzenie nowego tematu stanowi rozszerzenie tworzenia postu (funkcja *addTopic* wywołuje funkcję *addPost*) z tą różnicą, że nie można wykonać akcji (nie ma kogo atakować, ani przed kim się bronić).

4.2 Raporty

Backend projektu stworzony jest w PHP, z kolei frontend generowany w HTML. Backend zajmuje się przekazywaniem danych z formularzy do bazy (gdzie są odpowiednio przetwarzane) oraz generowaniem odpowiednich części strony (w zależności od wartości danych z bazy oraz tablicy _SESSION).
Dane w większości przypadków prezentowane są w postaci tabel, nawiązując do klasycznego stylu forum. Różnicę stanowi tutaj panel użytkownika z informacjami zebranymi w tzw. karcie.
W trakcie tworzenia projektu wykorzystano framework CSS Foundation oraz wtyczkę do wizualizacji danych ChartJS.

4.3 Panel administracyjny i makropolecenia

Aplikacja nie ma ściśle zdefiniowanego panelu administracyjnego. Zakłada się, że najwyższy administrator (twórca aplikacji) zarządza nią z poziomu bazy, w razie potrzeby wywołując odpowiednie kwerendy. Jednakże interweniuje on tylko w razie potrzeby (np. awaria).
Wyżsi rangą użytkownicy (administrator, moderator) są w stanie zobaczyć opcje ukryte dla zwykłych użytkowników, np. ukryte fora bądź przyciski odpowiadające za wymienione w punkcie 3.3.6 funkcjonalności. Pełnią one rolę panelu administracyjnego wbudowanego bezpośrednio w forum.

5 Dokumentacja

5.1 Wprowadzanie danych

Baza danych została wypełniona próbnymi danymi, w większości generowanym *Lorem ipsum*. Loginy dla użytkowników o poszczególnych rangach zostaną podane w załączniku (*logdata.txt*).

5.2 Dokumentacja użytkownika

1. Po wejściu na stronę pokazuje się wykaz forów. Możemy się po nich przemieszczać lub przejść do logowania bądź rejestracji. Po logowaniu kierowani jesteśmy na stronę główną, po rejestracji możemy się zalogować.
2. Można przeglądać tematy, posty bądź profile użytkowników poruszając się za pomocą linków. Zalogowani użytkownicy mogą przejrzeć swój profil, listę użytkowników albo przejść do wiadomości.
3. Przykładową bitwę można śledzić (bądź wziąć w niej udział po zalogowaniu) **w Koloseum, w temacie *Arena***. Można także atakować użytkowników w innych tematach lub pisać do nich wiadomości.
4. Jako moderator można usuwać posty wszystkich użytkowników i kierować ich (w tym siebie) na czarną listę. Taka autobanicja wiąże się z utratą uprawnień moderatora.
5. Jako administrator można usuwać tematy i posty.

5.3 Opracowanie dokumentacji technicznej

Dokumentacja wygenerowana przy pomocy *phpDox* zbajduje się w archiwum DokumentacjaPhp.zip

5.4 Załączniki

1. *table_struct.sql*
2. *functions.sql*
3. *logdata.txt*
4. *DokumentacjaPHP.zip*