Projekt - gra sieciowa

Przemysław Boroń
Wojciech Łupina
Jerzy Markiewicz
Przemysław Tekieli

Tematem projektu jest stworzenie gry typu MUD, razem z edytorem pozwalającym rozbudowywać świat gry o nowe elementy. Dodatkowo dostarczona zostanie również dokumentacja oraz testy pokrywające działanie systemu.

Lista zebranych wymagań:

- Gra ma zostać stworzona zgodznie z zasadami gier MUD-owych. Inspiracją może być gra MUME oraz reguły z gry ”Dungeons and Dragons”

Przyjęte wymagania na podstawie ogólnych zasad działania MUD-ów:

- gra ma korzystać z przeglądarki
- rozgrywka odbywa się poprzez wymienianie komunikatów tekstowych z serwerem
- gracz może poruszać się w 4 kierunkach
- na mapie mogą się znajdować przeszkody uniemożliwiające ruch z konkretnego pola w danym kierunku
- na mapie gracz może spotkać: przeciwników sterowanych przez komputer (mob-y), innych graczy oraz bohaterów niezależnych (NPC)
- gracz może atakować mob-y na mapie, które po swojej śmierci pozostawiają złoto i zwiększają doświadczenie gracza
- z jednym potworem w danym momencie może walczyć tylko jeden gracz
- po zdobyciu pewnej ilości doświadczenia gracz awansuje na wyższy poziom, co wiąże się ze zwiększeniem podstawowych statystyk
- zdobyty ekwipunek wpływa na statystyki gracza
- przedmioty można zakupić u niektórych NPC
- NPC mogą również dać graczowi misje do wykonania
- gdy gracz umrze to traci część zgromadzonego złota oraz doświadczenia
- po śmierci gracz trafia na z góry ustalone pole

Zebrane wymagania odnośnie projektu:

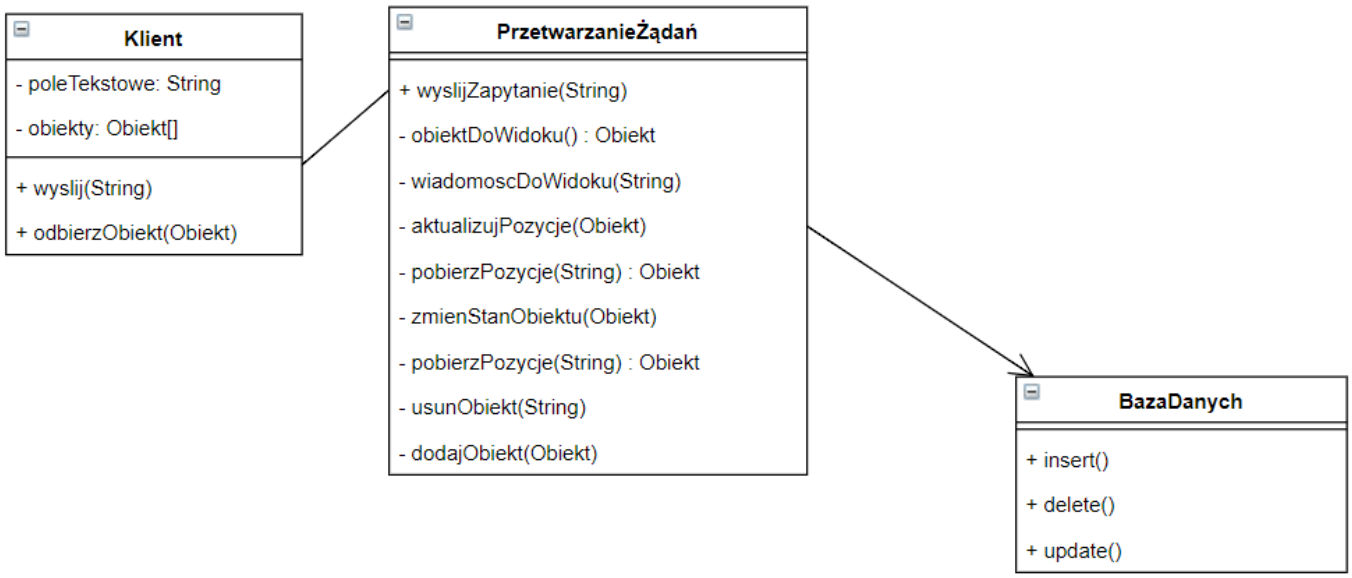
- oprócz gry ma zostać dostarczony edytor świata
- edytor pozwala na modyfikację gry w pełnym zakresie, można modyfikować m.in. parametry (jak np. ile doświadczenia traci się przy poziomie), ale także przeciwników, lokacje i misje, co pozwala np. tworzyć zupełnie nowych wrogów, i rozbudowywać obszar gry
- jeżeli byłoby dużo różnych parametrów do modyfikacji, to mogą się one znaleźć w oddzielnym pliku, w którym będą modyfikowane, ale główna część ma być modyfikowalna w edytorze graficznym
- są 3 rodzaje misji:
  - zabijanie potworów
  - podróż w pewne miejsce lub rozmowa z innym NPC
  - przyniesienie przedmiotu
- przeciwnicy dzielą się na agresywnych i pasywnych, np. przeciwnik ”krowa” będzie pasywna, za to ”pająk” będzie agresywny, czyli sam nas zaatakuje
- podstawowe statystyki identyfikujące postacie na mapie to: życie, mana, atak, obrona. Te statystyki zwiększają się przy zdobywaniu kolejnych poziomów doświadczenia przez postać
- ze względu na obecność w grze ”many” lub innego źródła energii magicznej, w grze będą występować podstawowe zaklęcia wykorzystujące manę

Architektura systemu

System będzie wykorzystywał znany wzorzec architektoniczny Model-Widok-Kontroler (Model-View-Controller) - MVC. W naszym systemie będą wyodrębnione 3 komponenty:

- interfejs użytkownika
- logika sterowania
- model danych

Zaletą jest odseparowanie części widocznej przez użytkownika od logiki systemu oraz sposobu przechowywania danych. System składa się z dwóch części, pierwszą z nich jest grywalna część, a drugą edytor. Dla części grywalnej:

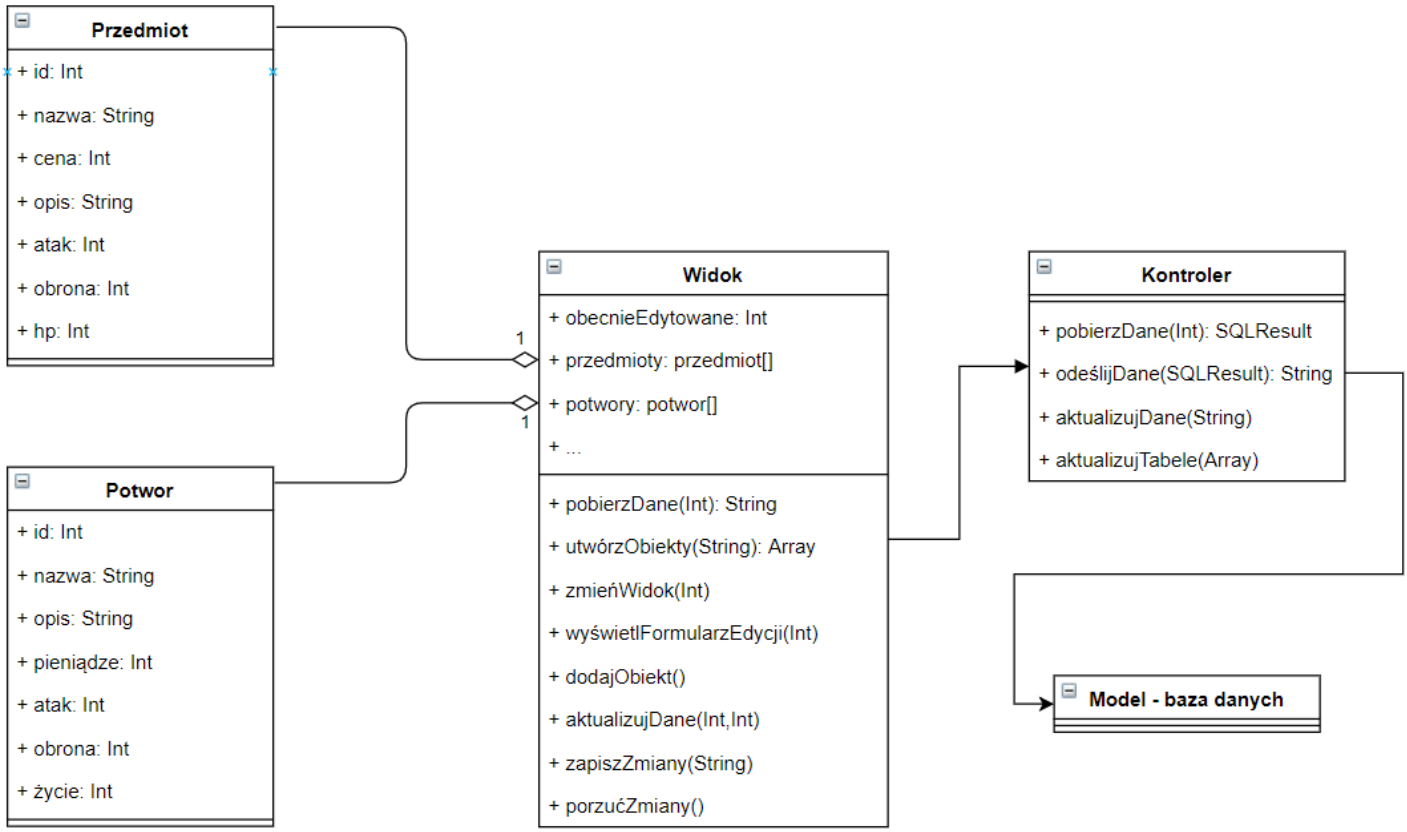


Widokiem jest część strony widoczna przez klienta, ”terminal” wyświetlający informacje otrzymane z serwera, i pozwalający wysyłać własne polecenia. Wiele widoków łączy się z tym samym kontrolerem.

Kontroler to nasz ”serwer”. Służy on za wysyłanie informacji do widoków, oraz za przetwarzanie otrzymach od nich poleceń w celu sprawdzenia czy polecenia są poprawne, a jeśli tak to prześle je do modelu, w formie dla niego zrozumiałej.

Modelem jest baza danych, w której znajdują się wszystko informacje o grze. Kontroler łączy się z nią i przesyła zapytania SQL, a baza zwraca rezultaty tych zapytań, które po przetworzeniu przez kontroler trafią do widoku.

Dla edytora:

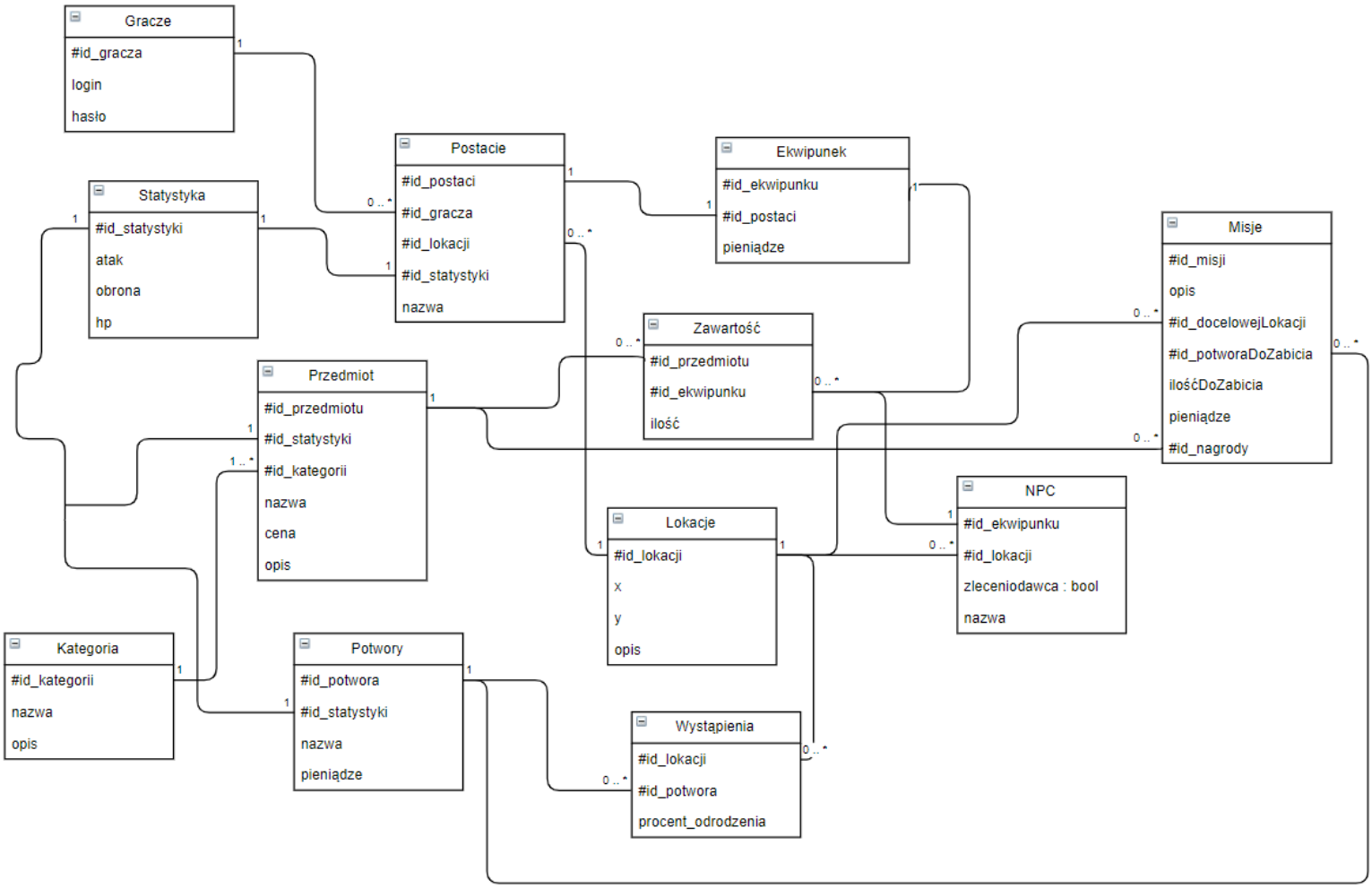


Widokiem jest edytor, wyświetlający edytowane dane, które można w tym miejscu zmieniać. Dane są pobierane z bazy danych, za pośrednictwem kontrolera, tak samo zmiany są wysyłane do kontrolera, który potem wyśle je do bazy danych.

Kontroler jest to pośrednik wymiany danych między widokiem a modelem, przede wszystkim zajmuje się przetwarzaniem danych otrzymanych od widoku w taki sposób żeby zrozumiał je model, oraz to samo od modelu do widoku.

Model to baza danych, dokładnie ta sama co w grywalnej części, na podstawie otrzymanych od kontrolera SQL-i wykonuje zadane czynności.

Wstępny projekt bazy danych:



Opis interfejsów:

Najważniejszym interfejsem jest ten łączący Widok z Kontrolerem w części grywalnej. Połączenie między tymi komponentami będzie realizowane przez socket-y. Kontroler będzie miał otwarty socket, który będzie akceptował połączenia z socketami z widoków. Dzięki połączeniu socketów możliwe będzie wysyałnie komunikatów tekstowym bez problemów w obie strony. Widok będzie wysyłał tekstowe komendy, który kontroler będzie sprawdzał w celu poprawności, a następnie odsyłał tekst będący informacją o błędzie w razie nie poprawnej komendy, lub wyśle tekstowo nową informację np. co się stało po wykonaniu ostatniej komendy. Połączenie socket-owe wydaje się być dobrym wyborem, ponieważ pozwala na komunikację klienta z serwrem, ale także na wysyłanie nowych informacji do klienta, bez wcześniejszego zapytania z jego strony.

Drugi interfejs łączy Widok z Kontrolerem w edytorze. Ze względu na to że wszystkie informacje pochodzące z kontrolera będą odpowiedzią na jakieś zapytanie z widoku (w przeciwieństwie do wcześniejszego interfejsu, nie może być sytuacji że widok otrzyma informacje ot tak), w tym miejscu wykorzystana będzie komunikacja AJAX. Widok będzie się komunikował z Kontrolerem za pomocą poleceń tekstowych oraz wykorzystując format JSON do przesyłania obiektów. Za każdym razem gdy Widok wyśle zapytanie AJAX do kontrolera, otrzyma od niego informacje zwrotną w postaci tekstu (ewentualnie pustego łancucha).

Połączenie kontrolera z modelem realizowane będzie z wykorzystaniem wbudowanych funkcji PHP, służących do łączenia się z bazą danych.

Lista wykorzystanych technologii (stos technologiczny):

- HTML5 - został wybrany ponieważ świetnie nadaje się do tworzenia stron internetowych i ogólnego front-end-u. Będzie on podstawą Widoku czyli strony Klienckiej, zostanie wykorzystany w miejscach widocznych dla użytkownika, czyli na stronie z ”terminalem”, oraz w edytorze.
- CSS3 - nieodłączna część html-a, pozwala ostylować elementy html-a, i do tego też posłuży. Wykorzystany tak jak html w miejscach widocznych dla użytkownika, czyli na stronie z ”terminalem”, oraz w edytorze.
- Javascript - razem z html-em i css-em zostanie użyty w front-end-zie. Główną rolą będzie obsługiwanie akcji użytkownika, to javascript pozwoli przy pomocy socket-u połączyć się z Kontrolerem, a także z jego pomocą będzie można wysyłać AJAX-y w edytorze. Na tym etapie nie przewidujemy użycia dodatkowych bibliotek lub framework-ów javascript-u, głównie z powodu nieznajomości takich, które mogłyby nam pomóc
- PHP - będzie odpowiedzialny za cały back-end, pośredniczy w wymianie informacji między użytkownikiem i bazą danych przy wykorzystaniu socket-u pozwoli na komunikację z użytkownikiem, a korzystanie z wbudowanych funkcji pozwoli na komunikację z bazą danych

Powyższe technologie zostaną wykorzystane przede wszystkim dlatego że wszyscy uczyliśmy się ich na studiach na 3 semestrze i wydają się wystarczyć do realizacji projektu

- PostgreSQL - popularny system do zarządzania bazą danych, akurat on zostanie wykorzystany, ponieważ wszyscy uczymy się go w tym semestrze

- Do komunikacji klient-serwer wykorzystamy technikę AJAX, pozwala ona na asynchroniczną wymianę informacji z serwerem bez przeładowywania całej strony. Technika ta zostanie wykorzystana przez edytor do komunikacji z serwerem, ponieważ wydaje się wystarczająca do przesyłania tekstów na serwer, oraz odbierania wiadomości zwrotnej z serwera (za sprawą formatu danych JSON, będziemy mogli wysyłać całe obiekty w formie tekstowej)
- Do komunikacji klient-serwer, serwer-klient wykorzystane zostaną sockety, które pozwalają na dwukierunkową wymianę informacji (tekstów), będzie to niezbędne przy grywalnej części, gdzie klient musi mieć możliwość wysyłania komend na serwer, ale jednocześnie musi mieć możliwość pobierania informacji od serwera, bez wysyłania wcześniejszego polecenia, ze znanych nam technik, można by również zastosować AJAX-y i wykorzystywać long-polling, ale ze względu na to że serwer może wysyłać często informacje do różnych klientów, lepiej mieć z nimi stałe połączenie, niż każdorazowo AJAX-owo tworzyć nowy poll

Projekt testów:

- Jednostkowe i integracyjne:
  - Javascript - JUnit, Jest
  - PHP - PHPUnit, Phake
- Systemowe i akceptacyjne:
  - Javascript - TestCafe
  - PHP - Behat

Lista narzędzi używanych przy realizacji projektu:
Cały projekt powstanie na Githubie z wykorzystaniem następujących narzędzi dostarczanych przez github-a:

- Repozytorium - ułatwi pracę nad jednym projektem wielu osobom jednocześnie
- Issue tracking & projekt managment tools - znacznie ułatwi zarządzanie kwestiami projektowymi, pozwali łatwo tworzyć nowe zgłoszenia błędów, i raportować statusy do poszczególnych zadań

- Code review - pozwala śledzić wszelkie zmiany w kodzie, co znacznie ułatwia pracę oraz ewentualne przywrócenei działającej wersji w przypadku dużych błędów