Programowanie aplikacji klient-serwer

Projekt 6 Egzamin testowy

Projekt

(wersja 1.1)

AUTORZY PROJEKTU:

Uladzislau Harbuz

Maksym Yakymyshyn

Spis treści

1 Wstęp

1.1 Treść zadania

Aplikacja umożliwia tworzenie i przeprowadzanie egzaminu testowego. 3 rodzaje użytkowników: administrator, egzaminator i student. Administrator tworzy grupy studenckie i przypisuje do nich studentów. Egzaminator wprowadza egzamin testowy (wszystkie pytania zamknięte) podając dla wszystkich pytań:

•Treść pytania,

•Propozycje odpowiedzi (co najwyżej 5),

•Poprawną odpowiedź (tylko 1).

Egzaminator przeprowadza egzamin udostępniając go wybranej grupie studentów na określony czas oraz uruchamia automatyczne sprawdzanie wyników testu dla poszczególnych studentów.

Po sprawdzeniu wynik testu jest przypisywany poszczególnym studentom.

Student loguje się, wprowadza odpowiedzi na poszczególne pytania i ogląda wyniki.

1.2 Dodatkowe założenia

1.2.1 Serwer

Serwer wykonany zostanie w języku „C++”. Dane o klientach, plikach i zamówieniach będą przechowywane w bazie danych

1.2.2 Klient

Klient będzie aplikacją napisaną w języku C#

2 Słownik pojęć

Sesja – abstrakcja połączenia zautentyfikowanego klienta z serwerem. Wszystkie komunikaty są wysyłane w ramach sesji. Wszystkie spróby komunikacji z serwerem poza aktywną sesją są odrzucane przez serwer.

3 Użyte protokoły

3.1 Protokół komunikacji klienta z serwerem

3.1.1 Wstęp

Został opracowany własny protokół komunikacji pomiędzy klientem a serwerem (Tablica 3.1.1.2.1).

3.1.1.1 Technologia

Zaprojektowana ramka danych składa się z:

Polecenie	Rozmiar danych	Dane

Polecenie – 1 bajt

Rozmiar danych – 1 bajt, który pokazuje rozmiar danych w sekcji „Dane”(dom. w bajtach).

W przypadku błędu po stronie serwera serwer musi zwracać komunikat ERROR_DATAGRAM (Tablica 3.1.1.2.1).

3.1.1.2 Funkcjonalność:

Protokół ten realizuje następującą funkcjonalność:

Tablica 3.1.1.2.1 – Opis protokołu komunikacji klient-serwer

Lp	Treść komunikatu	Spodziewana reakcja
1	GET_TEST_LIST_SIZE	Klient wysyła do serwera ramkę z polem „Rozmiar danych” = 0 i bez pola „Dane”. Klient otrzymuje liczbę testów w polu „Dane”.
2	GET_TEST	Klient w polu „Dane” wpisuje numer testu. Otrzymuje od serwera wypełnioną ramkę danych w formacie JSON zdefiniowanym następująco: „{‘text’:’tekst_pytania’, ‘variants’: [‘odp1’,’odp2’, …]}”.
3	SEND_TESTS_ANSWERS	Klient wysyła w polu „Dane” wybrane odpowiedzi dla testów w formacie JSON zdefiniowanym następująco: „{‘answewrs’:[‘test_n’:’ans_n’, …]}”. Klient dostaje w odpowiedź ramkę z kodem błędu.
4	OPEN_SESSION	Komunikat mówi serwerowi otworzyć nową sesję. Serwer czeka na nowe połączenie od klientów. Po uzyskaniu nowego połączenia serwer czeka na ten komunikat. Pole „Rozmiar danych” jest równe LOGIN_BYTE_SIZE + PASSWORD_BYTE_SIZE z tablicy 3.1.1.2.2, i w polu „Dane” do serwera są przesyłane login w pierwszych LOGIN_BYTE_SIZE i hasło w bajtach [LOGIN_BYTE_SIZE, PASSWORD_BYTE_SIZE].

		Klient dostaje w odpowiedź komunikat zwrotny z kodem błędu.
5	CLOSE_SESSION	Zamyka aktualną sesję. Kliend dostaje komunikat zwrotny z kodem błędu.
6	GET_RESULTS	Klient wysyła pustą ramkę. Dostaje od serwera ramke z polem „Dane” zawierającą wyniki testów w formacie JSON: „{‘pass’:’liczba_testów_zdanych’, ‘all’:’liczba_wszystkich_testów’}”
7	CHANGE_CREDENTIALS	Klient wpisuje w pole „Dane” w pierwsze LOGIN_BYTE_SIZE bajt nowy login, a w następne PASSWORD_BYTE_SIZE nowe hasło. Klient dostaje komunikat zwrotny z kodem błędu.
8	ADD_GROUP	Klient wpisuje w pole „Dane” nazwę nowej grupy. Dostaje od serwera komunikat zwrotny z kodem błędu.
9	ADD_USER	Klient wpisuje w pole danych : pierwsze LOGIN_BYTE_SIZE bajtów – login nowego użytkownika, w następne PASSWORD_BYTE_SIZE bajtów – nowe hasło nowego użytkownika, i w ostatnim bajcie wysyłą kod grupy nowego użytkownika z tablicy 3.1.1.2.3. Klient dostaje komunikat zwrotny z kodem błędu.
10	ERROR_DATAGRAM	Ten komunikat świadczy o tym, że w polu „Dane” w pierwszym bajcie znajduje się kod błędu , a w drugim bajcie - kod operacji której ten błąd dotyczy. Rozmiar pola „Dane” Wynosi 2 bajty.

Tablica 3.1.1.2.2 – Wartości stałe protokołu

Wartość	Nazwa	Opis
20	LOGIN_BYTE_SIZE	Rozmiar loginu w bajtach.
30	PASSWORD_BYTE_SIZE	Rozmiar hasła w bajtach.

3.1.2 Dokładny opis funkcjonalny poszczególnych metod/komend

3.1.2.1 XXXXXXXXXXXX (

Sygnatura:

Deklaracja (wartość zwracana, nazwa, parametry i ich znaczenie

Semantyka:

Po co jest / co robi

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Kody błędów:

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

3.1.3 Spis komunikatów błędów

Nr błędu	Komunikat
0x00	SUCCESS

3.2 Protokół przesyłania plików pomiędzy klientami(o ile występuje)

3.2.1 Wstęp

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

3.2.2 Żądanie

3.2.3 Odpowiedź

3.2.4 Spis komunikatów błędów

4 Przypadki użycia

4.1.1 Klient tworzy nowego użytkownika w bazie serwera

xxxxxxxxxXXXXXXXXXXXXXXXXXX Na przykład

4.1.2 Klient loguje się do serwera

Klient wywołuje login. Na przykład

4.1.3 Klient wylogowuje się z serwera

Klient wywołuje logout

4.1.4 XXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXX

5 Aplikacja klienta

5.1 Wstęp

Po co jest co robi.

5.2 Moduł nazwa1

Bardzo ogólnie funkcjonalność

5.3 Moduł nazwa2

Bardzo ogólnie funkcjonalność

6 Aplikacja serwera

6.1 Wstęp

Po co jest co robi.

6.2 Moduł nazwa1

Bardzo ogólnie funkcjonalność

6.3 Moduł nazwa2

Bardzo ogólnie funkcjonalność

7 Przechowywane dane

7.1 Wstęp

7.2 Tabela/ plik1

Po co jest, rodzaj przechowywanych danych

7.3 Tabela/ plik2

8 Interfejs danych

8.1 Funkcja 1

Krótki opis (po co, zwracana wartość, parametry)

8.2 Funkcja 2

Krótki opis (po co, zwracana wartość, parametry)

8.3 …

8.4 Spis komunikatów błędów

9 Planowany podział prac

W formie tabeli kto realizuje którą funkcję / moduł