



AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

69. Design and implement classes for DVD rental

z przedmiotu

Języki programowania obiektowego

Elektronika i Telekomunikacja 22/23

Mateusz Oleksy (416651)

Wtorek 15:00

prowadzący: Rafał Frączek

06.01.2025

1. Opis projektu

Projekt polega na zbudowaniu przy pomocy dostępnych narzędzi programistycznych poznanych na przedmiocie „Języki programowania obiektowego”, aplikacji symulującej wypożyczalnię DVD. Projekt realizowany za pomocą języka programowania o nazwie C++.

2. Project description

Projects itself represents project simulating DVD rental. Projects implements solutions and classes written in C++ language for proper DVD rental functionality.

3. Instrukcja użytkownika

Wchodząc do aplikacji ukazuje nam się wybór akcji: 0. Wyjście, 1. Logowanie, 2. Rejestracja. Opcję wybieramy poprzez numer przez całą aplikację, chyba że aplikacja wymaga podania innego typu danych. Po wybraniu odpowiedniej opcji, możemy dokonać logowania użytkownika poprzez podanie loginu, hasła. Możemy też przejść do rejestracji użytkownika. Program każe nam podać login i hasło (oczywiście inne niż istniejące w bazie), prawdziwe nazwisko, oraz unikalny kod PESEL, identyfikujący danego użytkownika. Możemy zalogować się na konto zwykłego użytkownika. Po zalogowaniu się trzeba potwierdzić dowolnym klawiszem i naszym oczom ukazuje się zbiór filmów oraz nasze dane w bazie danych klientów. Następnie mamy dostępne opcje 0. Wyjście, 1. Wyszukiwanie, 2. Wypożyczenie, 3. Zwrot filmu, 4. Zmiana loginu, hasła. Poprzez wybranie wyszukiwania możemy wyszukiwać danymi kryteriami filmy, wszystkie kryteria lub wybrane. Wypożyczenie przebiega na podaniu odpowiedniego ID filmów oraz liczby dni wypożyczenia. Przy zwrocie wpisujemy jakie ID filmu chcemy zwrócić. Zmiana loginu i hasła umożliwia zmianę danych logowania do bazy. Możemy zalogować się do systemu jako admin. W tym celu trzeba użyć danych, loginu: matole, hasła: 1111. Potem po potwierdzeniu loginu klawiszem, możemy wybrać dużo opcji. Ukazuje się nam widok baz danych oraz opcje 0. Wyjście, 1. Kliencie, 2. Filmy, 3. Wyszukiwanie. Po wybraniu opcji Klienci mamy do wyboru m.in. usunięcie konkretnego rekordu, dodanie rekordu (klienta), modyfikacja, wyszukiwanie tylko w bazie klientów. Należy przy tym wspomnieć, że przy modyfikacji klienta mamy jeszcze po wprowadzeniu trzech pól danych, sub-menu. W sub-menu możemy modyfikować, dodawać, usuwać, usuwać wszystkie filmy wypożyczone przez klientów. W menu Filmów możemy również dodawać, usuwać, modyfikować, wyszukiwać rekordy z bazy filmów. Mamy jeszcze ostatnią opcję czyli wyszukiwanie w obu bazach danych jednocześnie, po wprowadzeniu wyszukiwanej frazy. Warto dodać, że przy dodawaniu nowego klienta i filmu, pola nie mogą zostać puste. Jeżeli podczas modyfikacji rekordu klikniemy klawisz 'enter' to dana opcja pozostaje bez zmian. Jak widać użytkownik admin ma bardzo duże prawa i może praktycznie wszystko zmodyfikować. Zwykły użytkownik nie może wypożyczyć dwóch takich samych filmów. Admin może natomiast dodać dowolną ilość wypożyczeń. Dodatkowo admin może dodać użytkownika i jego hasło domyślne będzie 1111 oraz login ImięNazwisko. Użytkownik po dodaniu przez admina może zmienić hasło i login o ile tego chce.

4. Kompilacja

Kompilacja jest standardowa. Kompilacji podlega program o nazwie main.cpp. Ja kompilowałem poleceniem g++ main.cpp -o main.

5. Pliki źródłowe

W tym punkcie należy opisać wszystkie pliki źródłowe (.cpp, .h) w projekcie. Należy podać nazwę każdego pliku oraz informację o tym co się w nim znajduje. Na przykład:

Projekt składa się z następujących plików źródłowych:

- *main.cpp*
- *clients.h*, przechowującą funkcję oraz implementacje klas dla bazy klientów
- *movies.h*, przechowującą funkcję oraz implementacje klas dla bazy filmów
- *shared.h*, przechowującą funkcję pomocnicze dla plików źródłowych
- *settings.h*, przechowującą ustawienia kolorów dla powłoki

6. Zależności

- Program zawiera tylko jedną bibliotekę spoza STL, *windows.h* do wyświetlania kolorów w konsoli. (w *shared.h*)

7. Opis klas

W projekcie utworzono następujące klasy:

- *client* – reprezentuje rekord klienta
 1. `friend void add (vector <client>& tab);` - klasa dodająca nowych klientów
 2. `friend void reload (vector <client>& tab);` - klasa przeładowująca pliki dla klientów
 3. `void modify(vector <client>& tab, int n, vector <moviesClass::movie> movies);` - klasa modyfikująca rekordy klientów
 4. `void setId(const int idEnter)` – metoda ustawiająca pole ID
 5. `int getId(void)` - metoda pobierająca pole ID dla klienta
 6. `string getName ()` - metoda pobierająca pole Name (nazwa) dla klienta
 7. `string getSurname ()` - metoda pobierająca pole Surname (nazwisko) dla klienta
 8. `int getPesel ()` - metoda pobierająca pole Pesel (pesel) dla klienta
 9. `vector <string> getRentalDate ()` – metoda pobierająca wektor zawierający daty początku wypożyczenia dla klienta
 10. `vector <string> getDueDate ()` - metoda pobierająca wektor zawierający daty końca wypożyczenia dla klienta
 11. `vector <int> getMovieId ()` - metoda pobierająca wektor zawierający identyfikatory wypożyczonych filmów dla klienta
 12. `void setName (const string &nameEnter)` - metoda ustawiająca pole Name
 13. `void setSurname (const string &surnameEnter)` - metoda ustawiająca pole Surname
 14. `void setPesel (const int peselEnter)` - metoda ustawiająca pole Pesel
 15. `void setRentalDate (const vector <string> &rentalDateEnter)` - metoda ustawiająca pole Rental Date
 16. `void setDueDate (const vector <string> &dueDateEnter)` - metoda ustawiająca pole Due Date

17. void setterMovieId (const vector <int> &movieIdEnter) - metoda ustawiająca pole Movie ID
 18. void print() const; - metoda wyświetlająca informacje dla danego klienta
 19. void printTwo(); - metoda pomocnicza do tej powyżej, służy do wyświetlania informacji
 20. client(const int idEnter, const string &nameEnter, const string &surnameEnter, const int peselEnter, const vector <string> &rentalDateEnter, const vector <string> &dueDateEnter, const vector <int> &movieIdEnter) – konstruktor rekordu filmu
- movie – klasa zawierająca reprezentację rekordu filmu.
 1. friend int modify(vector <movie>& tab, int n); - klasa modyfikująca rekord filmu
 2. friend void add(vector <movie>& tab); - klasa dodająca rekord filmu
 3. friend void reload(vector <movie>& tab); - klasa przeładowująca pliki dla klasy movie
 4. movie(const int index, const string &titleEnter, const string &durationEnter, const string &dateEnter, const string &authorEnter, const string &genreEnter, const string &descriptionEnter, const int movieIdEnter) – konstruktor dla klasy movie
 5. void print() – metoda wyświetlająca dane
 6. void setterIndex(const int indexEnter) – metoda ustawiająca index filmu
 7. void setterTitle(const string &titleEnter) – metoda ustawiająca tytuł filmu
 8. void setterAuthor(const string &authorEnter) – metoda ustawiająca autora filmu
 9. void setterDuration(const string &durationEnter) – metoda ustawiająca czas trwania filmu
 10. void setterDate(const string &dateEnter) – metoda ustawiająca date wypuszczenia filmu
 11. void setterGenre(const string &genreEnter) – metoda ustawiająca gatunek filmu
 12. void setterDescription(const string &descriptionEnter) – metoda ustawiająca opis filmu
 13. void setterMovieId(const int movieIdEnter) – metoda ustawiająca identyfikator filmu
 14. int getIndex (void) – metoda pobierająca indeks filmu
 15. string getTitle(void) – metoda pobierająca tytuł filmu
 16. string getAuthor(void) – metoda pobierająca autora filmu
 17. string getDuration(void) – metoda pobierająca czas trwania filmu
 18. string getDate(void) – metoda pobierająca datę powstania filmu
 19. string getGenre(void) – metoda pobierająca gatunek filmu
 20. string getDescription(void) – metoda pobierająca opis filmu
 21. int getMovieId(void) – metoda pobierająca identyfikator filmu
 - Dodatkowe funkcje w namespace oraz pliku shared.h:
 1. setConsoleColor() – funkcja ustawiająca kolor terminala
 2. isNumeric() – funkcja sprawdzająca czy podana zmienna typu string jest liczbą
 3. stoiElem() – funkcja zamieniająca wektor elementów string na int
 4. tokenizer() – funkcja separująca dane w pliku oddzielone podanym separatorem

W projekcie wykorzystywane są następujące pliki zasobów:

- users.txt, struktura pliku:

w pliku po kolei oddzielone spacjami są dane klientów, czyli login hasło uprawnienia (admin/standard), pesel.

- clients.txt, struktura pliku:

pierwsza linia zawiera nazwy widoczne w pierwszym wierszu bazy danych, następnie w kolejnych wierszach pola wchodzące w skład struktury rekordu klienta.

- movies.txt, struktura pliku:

pierwsza linia zawiera nazwy widoczne w pierwszym wierszu bazy danych, następnie w kolejnych wierszach pola wchodzące w skład struktury rekordu filmu.

Warto dodać, że wszystkie dane z sobą wiążą praktycznie dwa istotne pola zawierające się w nich. Jest to pesel oraz movieID. Pesel służy do identyfikacji klientów oraz ich powiązaniu login/hasła z realnymi danymi. Natomiast dane wypożyczenia są powiązane za pomocą movieID. Jeżeli np. zmienimy movieID to wypożyczenia klientów dla tego samego filmu też się zaktualizują itd.

8. Dalszy rozwój i ulepszenia

Gdyby przenieść funkcjonalność na serwer, to do dyspozycji jest rozbudowana aplikacja do obsługi klientów oraz zarządzania wypożyczalnią. W przyszłości można dodać by system płatności. Mimo wszystko projekt był dość pracochłonny i wymagający, kod z wszystkich plików liczy ponad 1583 linijek kodu.

9. Inne

brak