Politechnika Śląska w Gliwicach Wydział Automatyki, Elektroniki i Informatyki.

Programowanie Komputerów 4

System obsługi biblioteki

Autor	Jagoda Chmielewska
Prowadzący	mgr inż. Grzegorz Kwiatkowski
Kierunek	informatyka
Rodzaj studiów	SSI
Semestr	4
Termin laboratorium	wtorek 12:00-12:30
Grupa	1
Data oddania sprawozdania	02.06.2020

Repozytorium projektu: https://github.com/jagodach/PROJEKT-PK4-BIBLIOTEKA

1. Temat

Zadanie polegało na napisaniu programu w języku C++ z wykorzystaniem zagadnień poznanych na laboratorium. Wybrany przeze mnie program jest systemem zarządzania/obsługi biblioteki. Dane przechowywane są w plikach csv. Pliki te zachowują się podobnie do plików txt. Format danych jest ważny tylko dla zapisywania i pobierania danych.

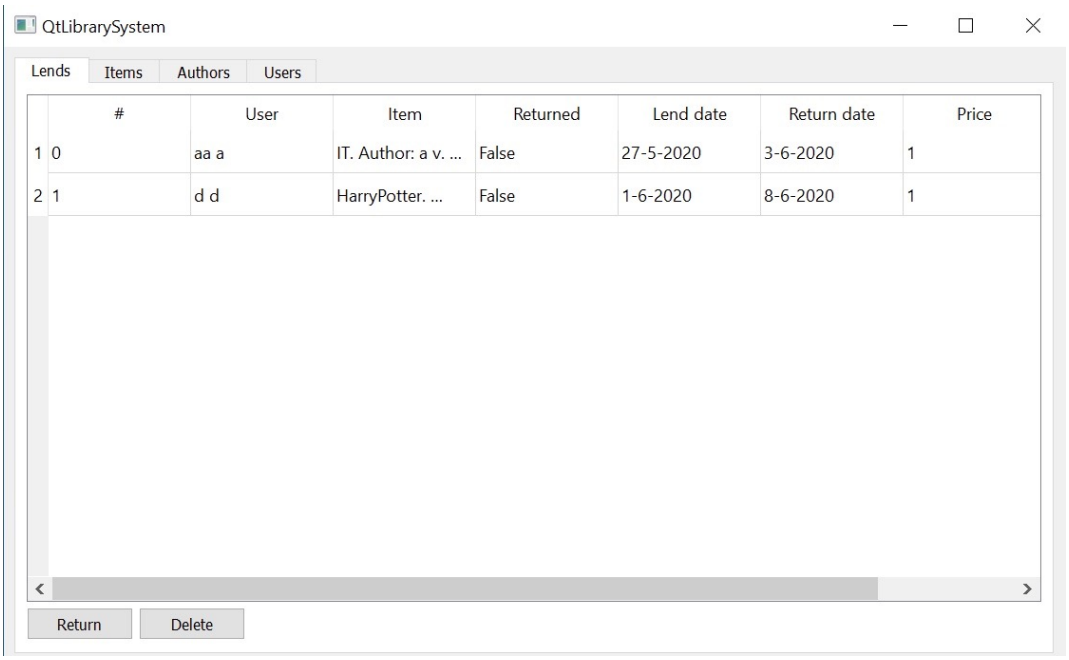
2. Analiza tematu

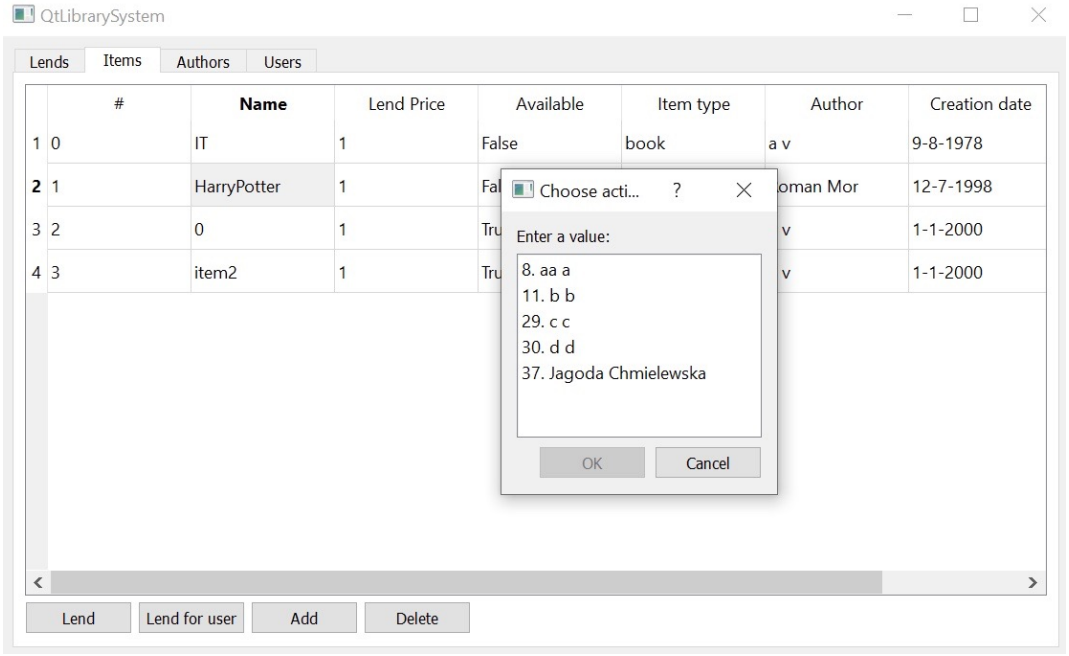
Przystępując do analizy zadania starałam się brać pod uwagę funkcjonalność i optymalność programu, jak i jak najlepsze doświadczenie dla użytkownika. W związku z tym logika programu jest podatna na rozbudowy, a interfejs starałam się żeby nie „krzyczał" kolarami i był przyjemny w odbiorze dla użytkownika biblioteki. Różne typy obiektów są przechowywane w innych wektorach. Wektory posiadają wskaźniki do obiektów. Obiekty posiadają referencje do siebie nawzajem. Przechowują tylko id w plikach z danymi. Używają ID żeby znaleźć obiekt i związać go z innymi kolekcjami. Po zalogowaniu się jako user czyli zwykły użytkownik możemy tylko zobaczyć aktualne wypożyczenia danego użytkownika na którego jesteśmy zalogowani, zobaczyć historyczne wypożyczenia danego użytkownika, wypożyczyć przedmioty (książki papierowe, ebooki), zwrócić przedmioty(książki papierowe, ebooki). Po zalogowaniu się jako admin jest dostępnych dużo więcej funkcji. Użytkownik musi zapłacić za wypożyczenie (cena wypożyczenia + 10% za każdy tydzień opóźnienia). Użytkownik ma także oczywiście możliwość opuszczenia programu bibliotecznego. Admin może zobaczyć aktualne wypożyczenia wszystkich użytkowników biblioteki, zobaczyć historyczne wypożyczenia wszystkich użytkowników biblioteki, może wypożyczyć książkę dla wybranego przez siebie użytkownika, zwrócić książkę dla wybranego przez siebie użytkownika, dodawać autorów, usuwać autorów, dodawać przedmioty (książki papierowe, ebooki), usuwać przedmioty (książki papierowe, ebooki) oraz dodawać dane o nich np rok powstania, rok wydania, format ebooka itp. oraz ma możliwość wyjścia z programu. Każdy przedmiot może być wypożyczony raz (czyli jeśli chcemy mieć w bibliotece trzy te same książki to muszą zostać one dodane trzy razy).

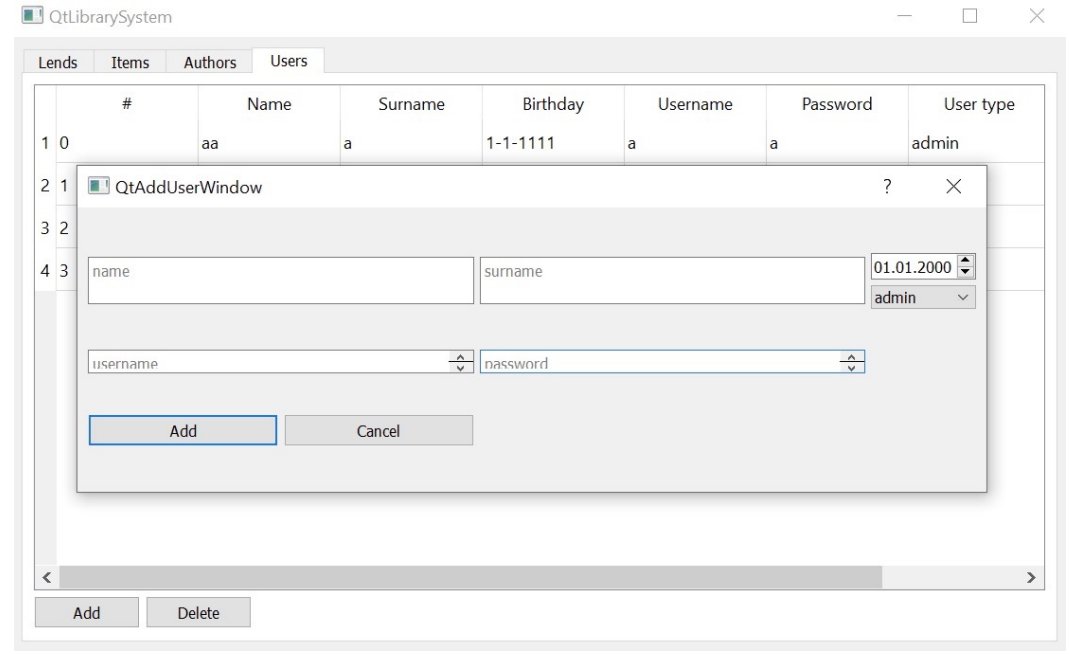
3. Specyfikacja zewnętrzna

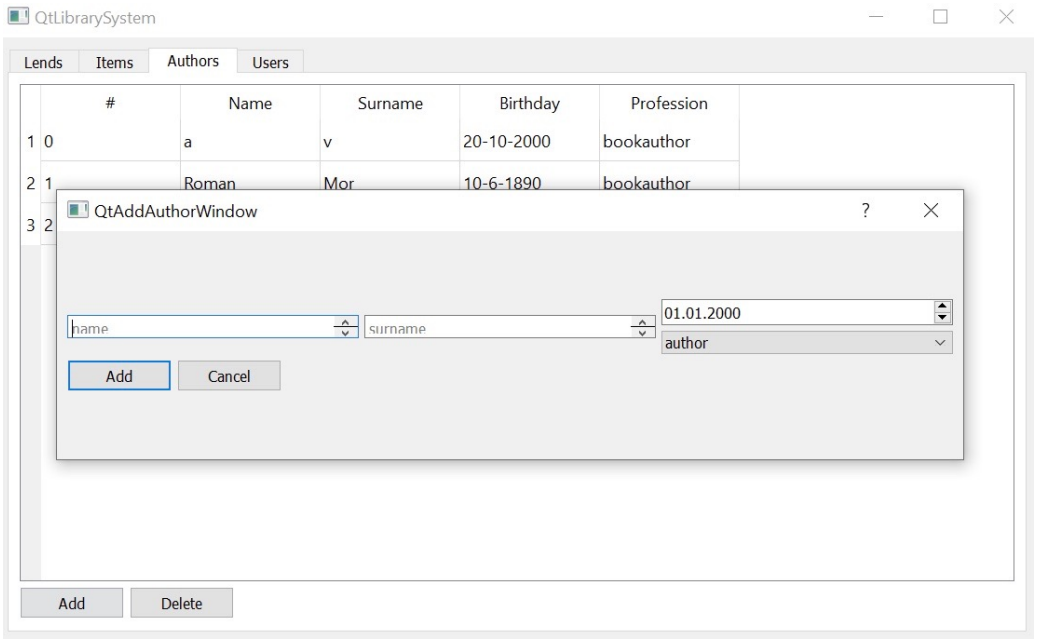
Program korzysta z interfejsu graficznego stworzonego przy pomocy QT.

Screeny aplikacji widoku dostępnego dla admina:

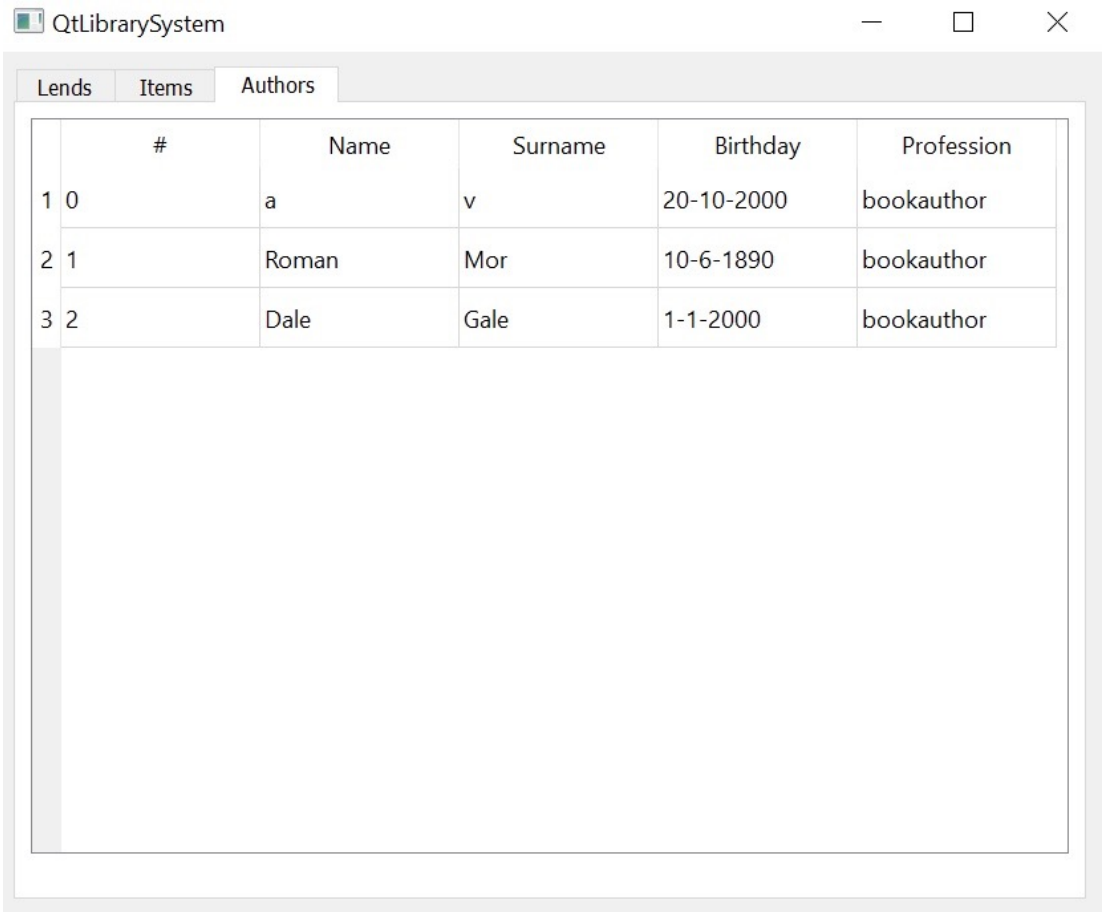

QtLibrarySystem
Lends
Items
Authors
Users
User Item Returned Lend date Return date Price
1 0 aa a IT. Author: a v. ... False 27-5-2020 3-6-2020 1
2 1 d d HarryPotter. ... False 1-6-2020 8-6-2020 1
Return
Delete

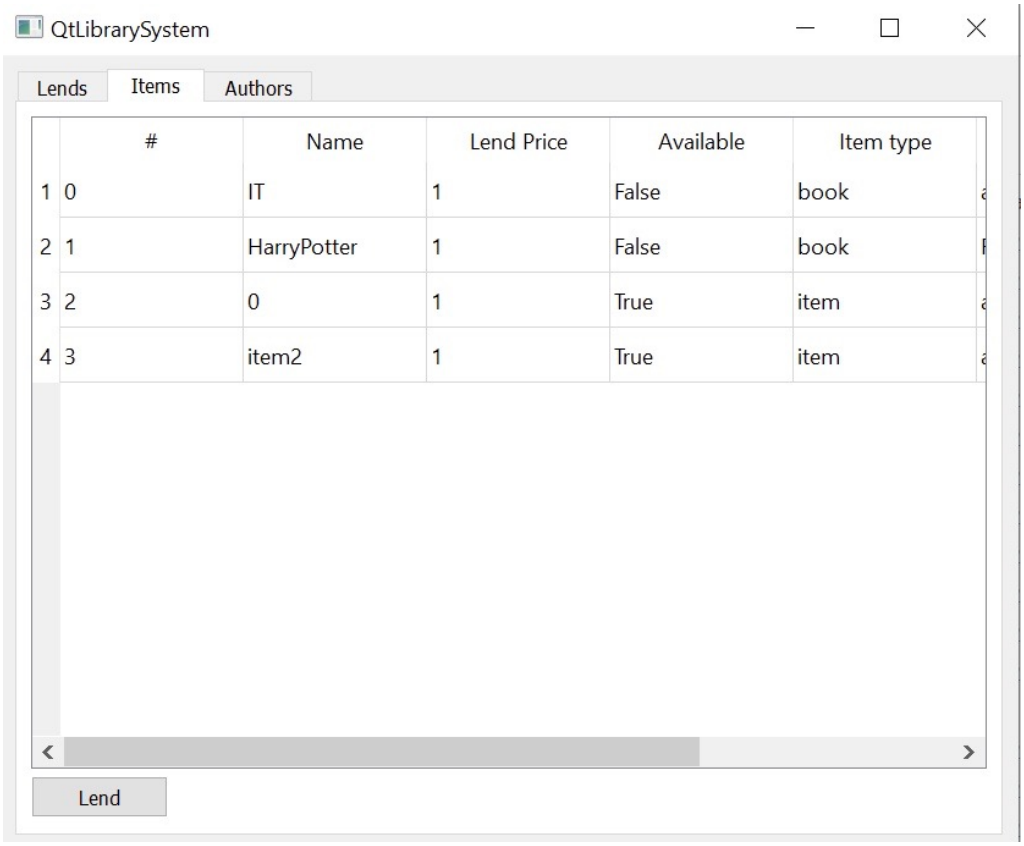

QtLibrarySystem
Lends
Items
Authors
Users
Name Lend Price Available Item type Author Creation date
1 0 IT 1 False book a v 9-8-1978
2 1 HarryPotter 1 oman Mor 12-7-1998
3 2 0 1 v 1-1-2000
4 3 item2 1 v 1-1-2000
Choose acti... ?
Enter a value:
8. aa a
11. b b
29. c c
30. d d
37. Jagoda Chmielewska
OK
Cancel
Lend
Lend for user
Add
Delete

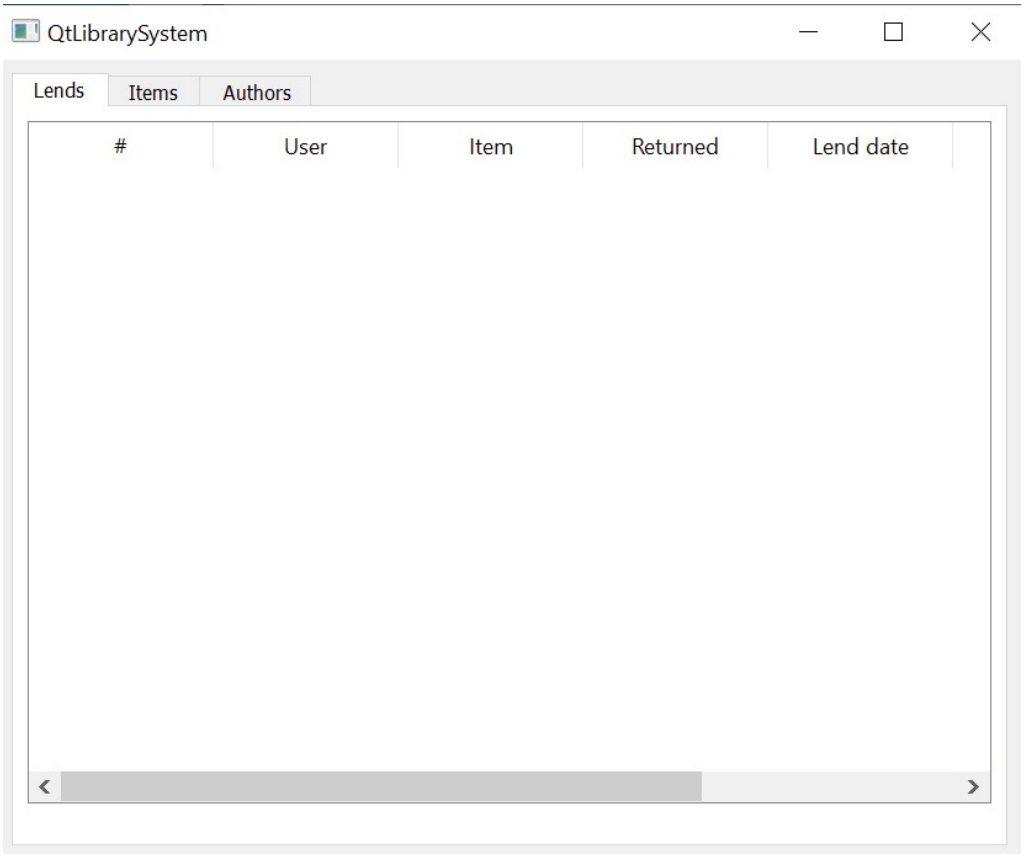

QtLibrarySystem
Lends
Items
Authors
Users
Name Surname Birthday Username Password User type
1 0 aa a 1-1-1111 a a admin
QtAddUserWindow ?
name
surname
01.01.2000
admin
username
password
Add
Cancel
Add
Delete



Screeny aplikacji widoku dostępnego dla usera:







4. Specyfikacja wewnętrzna

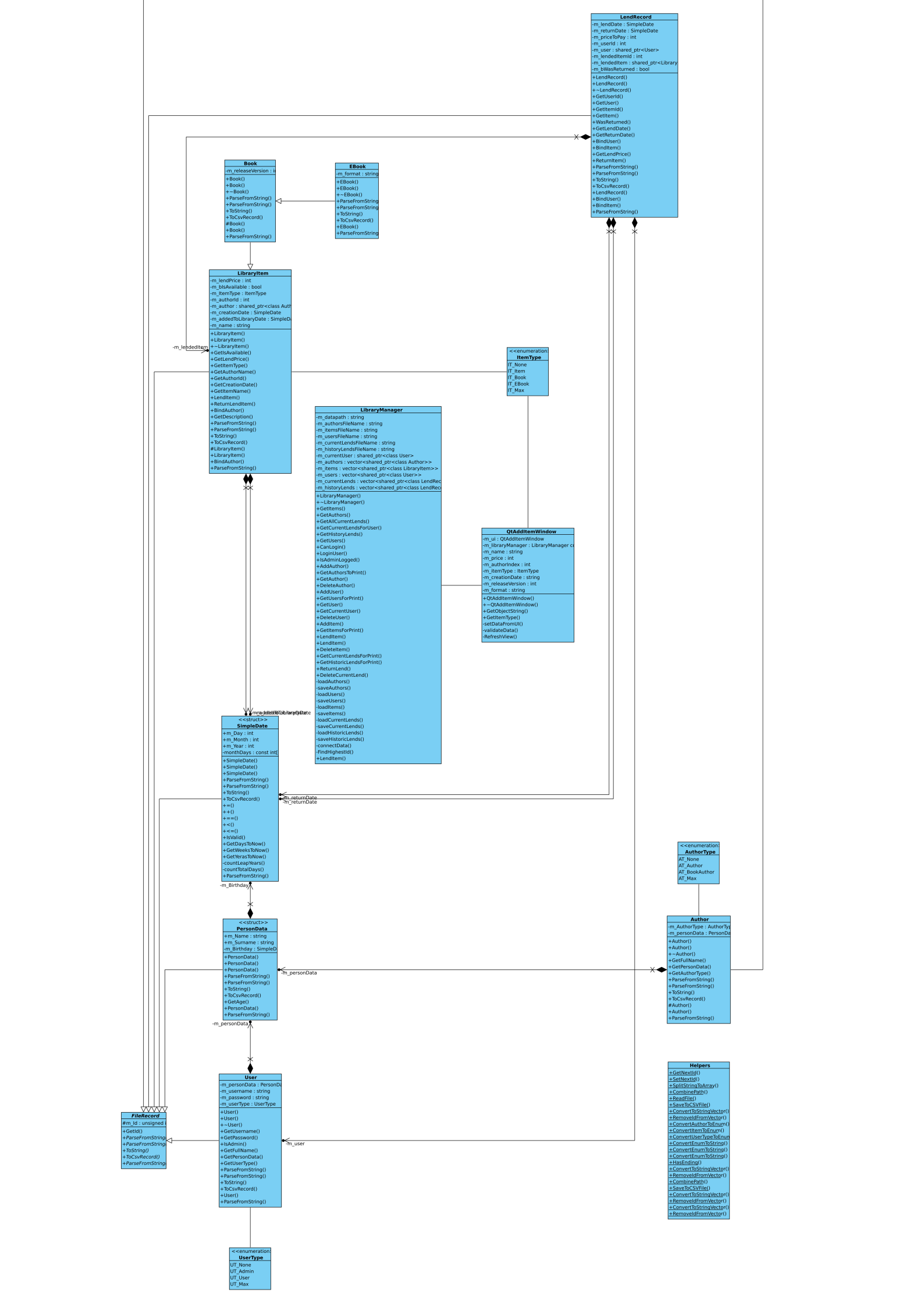

LendRecord
-m_lendDate : SimpleDate
-m_returnDate : SimpleDate
-m_priceToPay : int
-m_userId : int
-m_user : shared_ptr<User>
-m_lendedItemId : int
-m_lendedItem : shared_ptr<Library
-m_bWasReturned : bool
+LendRecord()
+LendRecord()
+~LendRecord()
+GetUserId()
+GetUser()
+GetItemId()
+GetItem()
+WasReturned()
+GetLendDate()
+GetReturnDate()
+BindUser()
+BindItem()
+GetLendPrice()
+ReturnItem()
+ParseFromString()
+ParseFromString()
+ToString()
+ToCsvRecord()
+LendRecord()
+BindUser()
+BindItem()
+ParseFromString()
Book
-m_releaseVersion : i
+Book()
+Book()
+~Book()
+ParseFromString()
+ParseFromString()
+ToString()
+ToCsvRecord()
#Book()
+Book()
+ParseFromString()
EBook
-m_format : string
+EBook()
+EBook()
+~EBook()
+ParseFromString
+ParseFromString
+ToString()
+ToCsvRecord()
+EBook()
+ParseFromString
LibraryItem
-m_lendPrice : int
-m_bIsAvailable : bool
-m_ItemType : ItemType
-m_authorId : int
-m_author : shared_ptr<class Auth
-m_creationDate : SimpleDate
-m_addedToLibraryDate : SimpleDa
-m_name : string
+LibraryItem()
+LibraryItem()
+~LibraryItem()
+GetIsAvailable()
+GetLendPrice()
+GetItemType()
+GetAuthorName()
+GetAuthorId()
+GetCreationDate()
+GetItemName()
+LendItem()
+ReturnLendItem()
+BindAuthor()
+GetDescription()
+ParseFromString()
+ParseFromString()
+ToString()
+ToCsvRecord()
#LibraryItem()
+LibraryItem()
+BindAuthor()
+ParseFromString()
-m_lendedItem
<<enumeration>>
ItemType
IT_None
IT_Item
IT_Book
IT_EBook
IT_Max
LibraryManager
-m_datapath : string
-m_authorsFileName : string
-m_itemsFileName : string
-m_usersFileName : string
-m_currentLendsFileName : string
-m_historyLendsFileName : string
-m_currentUser : shared_ptr<class User>
-m_authors : vector<shared_ptr<class Author>>
-m_items : vector<shared_ptr<class LibraryItem>>
-m_users : vector<shared_ptr<class User>>
-m_currentLends : vector<shared_ptr<class LendRec
-m_historyLends : vector<shared_ptr<class LendRec
+LibraryManager()
+~LibraryManager()
+GetItems()
+GetAuthors()
+GetAllCurrentLends()
+GetCurrentLendsForUser()
+GetHistoryLends()
+GetUsers()
+CanLogin()
+LoginUser()
+IsAdminLogged()
+AddAuthor()
+GetAuthorsToPrint()
+GetAuthor()
+DeleteAuthor()
+AddUser()
+GetUsersForPrint()
+GetUser()
+GetCurrentUser()
+DeleteUser()
+AddItem()
+GetItemsForPrint()
+LendItem()
+LendItem()
+DeleteItem()
+GetCurrentLendsForPrint()
+GetHistoricLendsForPrint()
+ReturnLend()
+DeleteCurrentLend()
-loadAuthors()
-saveAuthors()
-loadUsers()
-saveUsers()
-loadItems()
-saveItems()
-loadCurrentLends()
-saveCurrentLends()
-loadHistoricLends()
-saveHistoricLends()
-connectData()
-FindHighestId()
+LendItem()
QtAddItemWindow
-m_ui : QtAddItemWindow
-m_libraryManager : LibraryManager c
-m_name : string
-m_price : int
-m_authorIndex : int
-m_itemType : ItemType
-m_creationDate : string
-m_releaseVersion : int
-m_format : string
+QtAddItemWindow()
+~QtAddItemWindow()
+GetObjectString()
+GetItemType()
-setDataFromUI()
-validateData()
-RefreshView()
<<struct>>
SimpleDate
+m_Day : int
+m_Month : int
+m_Year : int
-monthDays : const int[
+SimpleDate()
+SimpleDate()
+SimpleDate()
+ParseFromString()
+ParseFromString()
+ToString()
+ToCsvRecord()
+=()
++()
+==()
+<()
+<=()
+IsValid()
+GetDaysToNow()
+GetWeeksToNow()
+GetYerasToNow()
-countLeapYears()
-countTotalDays()
+ParseFromString()
-m_returnDate
-m_returnDate
-m_Birthday
<<enumeration>>
AuthorType
AT_None
AT_Author
AT_BookAuthor
AT_Max
<<struct>>
PersonData
+m_Name : string
+m_Surname : string
-m_Birthday : SimpleD
+PersonData()
+PersonData()
+PersonData()
+ParseFromString()
+ParseFromString()
+ToString()
+ToCsvRecord()
+GetAge()
+PersonData()
+ParseFromString()
-m_personData
-m_personData
Author
-m_AuthorType : AuthorTyp
-m_personData : PersonDa
+Author()
+Author()
+~Author()
+GetFullName()
+GetPersonData()
+GetAuthorType()
+ParseFromString()
+ParseFromString()
+ToString()
+ToCsvRecord()
#Author()
+Author()
+ParseFromString()
FileRecord
#m_Id : unsigned
+GetId()
+ParseFromString
+ParseFromString
+ToString()
+ToCsvRecord()
+ParseFromString
User
-m_personData : PersonDa
-m_username : string
-m_password : string
-m_userType : UserType
+User()
+User()
+~User()
+GetUsername()
+GetPassword()
+IsAdmin()
+GetFullName()
+GetPersonData()
+GetUserType()
+ParseFromString()
+ParseFromString()
+ToString()
+ToCsvRecord()
+User()
+ParseFromString()
-m_user
Helpers
+GetNextId()
+SetNextId()
+SplitStringToArray()
+CombinePath()
+ReadFile()
+SaveToCSVFile()
+ConvertToStringVector()
+RemoveIdFromVector()
+ConvertAuthorToEnum()
+ConvertItemToEnum()
+ConvertUserTypeToEnum
+ConvertEnumToString()
+ConvertEnumToString()
+ConvertEnumToString()
+HasEnding()
+ConvertToStringVector()
+RemoveIdFromVector()
+CombinePath()
+SaveToCSVFile()
+ConvertToStringVector()
+RemoveIdFromVector()
+ConvertToStringVector()
+RemoveIdFromVector()
<<enumeration>>
UserType
UT_None
UT_Admin
UT_User
UT_Max

Diagram klas: https://github.com/jagodach/PROJEKT-PK4-BIBLIOTEKA/blob/master/Class%20Diagram2.svg

Specyfikacja klas:

Przykłady elementów omawianych na labaratorium znajdujących się w projekcie:

Algorytm biblioteki STL - przykład użycia (używane są wielokrotnie także w innych przypadkach)

```
1.  template<class T>
2.  void Helpers::SaveToCSVFile(std::vector<T> recordsToSave,
    std::filesystem::path pathToFile)
3.  {
4.        std::ofstream fileToSave(pathToFile);
5.
6.        std::for_each(recordsToSave.begin(), recordsToSave.end(),
7.              [&fileToSave](const auto element)
8.        {
9.              fileToSave << element->ToCsvRecord() << std::endl;
10.       });
11.
12.       fileToSave.close();
13. }
14.
15. template<class T>
16. static std::vector<std::string>
17.       Helpers::ConvertToStringVector(std::vector<T>
    recordsToConvert)
```

```
18. {
19.       std::vector<std::string> records;
20.
21.       std::for_each(recordsToConvert.begin(),
    recordsToConvert.end(),
22.             [&records](const auto element)
23.       {
24.             records.push_back(element->ToString());
25.       });
26.
27.       return records;
28. }
```

Szablon(template) - przykład użycia (używane są wielokrotnie także w innych przypadkach)

```
1.  // Template to save FileRecord vector to file.
2.       // It allows to not read many times the same code -
    compiler is doing it for us.
3.       template<class T>
4.       static void SaveToCSVFile(std::vector<T> recordsToSave,
    std::filesystem::path pathToFile);
5.       // Template to convert FileRecords vector to display
    strings vector.
6.       template<class T>
7.       static std::vector<std::string>
    ConvertToStringVector(std::vector<T> recordsToConver);
8.       // Template to remove FileRecord from vector.
9.       // Vector is passed as reference (&) - it means that we
    pass address for vector and we use the same vector as the one
```

```
    passed (it's the same object, not only copied value)
10.     template<class T>
11.     static bool RemoveIdFromVector(std::vector<T>&
    recordsToConver, int id);
```

Regex

Kontenery STL - przykład użycia (używane są wielokrotnie także w innych przypadkach)

```
User(std::vector<std::string> stringToParse);

User(std::vector<std::string> stringToParse);virtual void
ParseFromString(std::vector<std::string> stringVector) override
```

Smartpointery - przykład użycia (używane są wielokrotnie także w innych przypadkach)

```
1.  namespace LibrarySystem::Globals
2.  {
3.  LibraryManager* LibraryManager;
4.  }
```

```
1.  SimpleDate operator= (SimpleDate dateToSet)
2.      {
3.          m_Day = dateToSet.m_Day;
4.          m_Month = dateToSet.m_Month;
5.          m_Year = dateToSet.m_Year;
6.          return *this;
```

```
1.  bool operator<= (const SimpleDate& objectToCompare)
2.      {
```

```
3.            return *this == objectToCompare || *this <
    objectToCompare;

4.        }
```

5. Testowanie i uruchamianie

Program działa poprawnie, nie zostały wykryte żadne błędy. Program można uruchomić z pliku .exe lub normalnie po kompilacji w visualu. Wyświetla się okno startowe gdzie należy się zalogować, aby móc korzystać z przypisanych do danego konta funkcji. Są one różne dla usera i dla admina.

6. Wnioski

Zrealizowanie tego projektu umożliwiło mi skonstruowanie programu z interfejsem graficznym wykonanym w QT, pozwalającym na wygodną interakcję użytkownika z programem. Problem oddzielenia interfejsu od logiki oraz optymalizacja silnika gry zdecydowanie rozwinął moją umiejętność tworzenia oprogramowania. Skorzystanie z zagadnień omawianych na laboratorium pozwoliło mi dzięki użyciu kontenerów na wygodne przechowywanie danych. Użyte zostały także smartpointery, algorytmy STL, regex oraz szablony. Myślę, że udało mi się zapewnić przyjemną obsługę programu oraz bardzo wygodny, podatny na rozbudowę silnik.