|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ ИУ6

КАФЕДРА Компьютерные системы и сети

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ №3 (Вариант 13)**

Студент Костарев Кирилл Павлович

*фамилия, имя, отчество*

Группа ИУ6-24Б

Тип практики Практика по получению первичных профессиональных умений и навыков (Программирование)

Название предприятия МГТУ им. Н.Э. Баумана

Студент  **\_\_\_\_\_\_\_ Костарев Кирилл Павлович**

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2019 г.*

**Оглавление отчета**

1. Условие задания. Цель практикума. Задача практикума. Код……………….3
2. Диаграмма объектов …………………………………………………………. 18
3. Диаграмма класса……………………………………………………………...19
4. Формы интерфейса…………………………………………………………….20
5. Диаграмма состояний интерфейса……………………………………………22
6. Диаграмма последовательностей……………………………………………..23
7. Вывод…………………………………………………………………………...24

**Создание программной системы c Qt интерфейсом на С++.**

**Условие задания:**

Выполнить объектную декомпозицию, разработать формы интерфейса, диаграмму состояний интерфейса, диаграммы классов интерфейсной и предметной областей, диаграмму последовательностей одной из реализуемых операций. Разработать, протестировать и отладить программу в среде Visual Studio или QT Creator.

В электронной картотеке библиотеки о каждой книге хранится информация: автор, название, издательство, год издания, число экземпляров. Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

1. Получить названия всех книг данного автора.

2. Получить фамилии авторов тех книг, количество экземпляров которых не меньше заданного.

3. Определить названия всех книг тех авторов, которые издавались с данного года в данном издательстве.

4. Построить диаграмму распределения книг по годам издания.

**Целью практикума** является получение навыков создания небольших программных систем с оконными и консольными интерфейсами.

**Задачи практикума являются:**

- Овладение методикой и получение практических навыков проектирования небольших программных систем при структурном и объектном подходах;

- более глубокое изучение средств реализации проектов программ на одном из изучаемых универсальных языках программирования высокого уровня;

- воспитание внимания, аккуратности, систематичности, а также формирование интереса к изучаемой профессиональной деятельности.

**Код модуля “Главная форма приложения”:**

1. #ifndef MAINFORM\_H
2. #define MAINFORM\_H
3. #include <QtGui>
4. #include <QWidget>
5. #include <QPushButton>
6. #include "addform.h"
7. #include "printform.h"
8. #include "findauthorsform.h"
9. #include "findauthorsbynumbersofbooksform.h"
10. #include "findauthorsbypublishingandyearform.h"
11. #include "widget.h"

14. **class** Window: **public** QWidget
15. {
16. Q\_OBJECT
17. QPushButton \*btnAdd, \*btnFindAuthors, \*btnFindAuthorsByNumbersOfBooks,
18. \*btnFindAuthorsByPublishingAndYear, \*btnDiagram,
19. \*btnDeleteData, \*btnExit;
20. addForm winAdd;
21. findAuthorsForm winFindAuthors;
22. findAuthorsByNumbersOfBooksForm winFindAuthorsByNumbersOfBooks;
23. findAuthorsByPublishingAndYearForm winFindAuthorsByPublishingAndYear;
24. Widget winWidget;
25. **public**:
26. Window();
27. **public** slots:
28. **void** showAdd();
29. **void** showFindAuthors();
30. **void** showFindAuthorsByNumbersOfBooks();
31. **void** showFindAuthorsByPublishingAndYear();
32. **void** showWidget();
33. **void** DeleteData();
34. };
36. #endif // MAINFORM\_H
37. #include "mainform.h"
38. #include <QPushButton>
39. #include <QWidget>
40. #include <QMessageBox>
41. #include <QVBoxLayout>
42. Window::Window()
43. {
44. **this**->setWindowTitle("Electronic Library");
45. btnAdd = **new** QPushButton("Create/Add Data About Author", **this**);
46. btnFindAuthors = **new** QPushButton("Find All Books Of Authors", **this**);
47. btnFindAuthorsByNumbersOfBooks = **new** QPushButton("Find Authors By Numbers Of Books", **this**);
48. btnFindAuthorsByPublishingAndYear = **new** QPushButton("Find Authors By Current Publishing And Current Year",**this**);
49. btnDiagram = **new** QPushButton("Diagram Of Distribution Of Books",**this**);
50. btnDeleteData = **new** QPushButton("Delete Data",**this**);
51. btnExit = **new** QPushButton("Exit", **this**);
53. QVBoxLayout \*layout = **new** QVBoxLayout(**this**);
54. layout->addWidget(btnAdd);
55. layout->addWidget(btnFindAuthors);
56. layout->addWidget(btnFindAuthorsByNumbersOfBooks);
57. layout->addWidget(btnFindAuthorsByPublishingAndYear);
58. layout->addWidget(btnDiagram);
59. layout->addWidget(btnDeleteData);
60. layout->addWidget(btnExit);
62. resize(450,400);
64. connect(btnAdd, SIGNAL(clicked()), **this**, SLOT(showAdd()));
65. connect(btnFindAuthors, SIGNAL(clicked()), **this**, SLOT(showFindAuthors()));
66. connect(btnFindAuthorsByNumbersOfBooks, SIGNAL(clicked()), **this**, SLOT(showFindAuthorsByNumbersOfBooks()));
67. connect(btnFindAuthorsByPublishingAndYear, SIGNAL(clicked()), **this**, SLOT(showFindAuthorsByPublishingAndYear()));
68. connect(btnDiagram, SIGNAL(clicked()), **this**, SLOT(showWidget()));
69. connect(btnDeleteData, SIGNAL(clicked()), **this**, SLOT(DeleteData()));
70. connect(btnExit, SIGNAL(clicked()), **this**, SLOT(close()));
71. }
73. **void** Window::showAdd()
74. {
75. winAdd.show();
76. }
78. **void** Window::showFindAuthors()
79. {
80. winFindAuthors.show();
81. }
83. **void** Window::showFindAuthorsByNumbersOfBooks()
84. {
85. winFindAuthorsByNumbersOfBooks.show();
86. }
88. **void** Window::showFindAuthorsByPublishingAndYear()
89. {
90. winFindAuthorsByPublishingAndYear.show();
91. }
93. **void** Window::showWidget()
94. {
95. winWidget.show();
96. }
98. **void** Window::DeleteData()
99. {
100. QFile("book.txt").remove();
101. }

**Код модуля “Добавление информации об авторе”:**

1. #ifndef ADDFORM\_H
2. #define ADDFORM\_H
3. #include <QtGui>
4. #include <QWidget>
5. #include <QLabel>
6. #include <QPushButton>
7. #include <QLineEdit>
8. **class** addForm : **public** QWidget
9. {
10. Q\_OBJECT
11. QLabel \*author, \*name, \*publisher, \*yearOfPublication, \*numberOfCopies;
12. QLineEdit \*authorEdit, \*nameEdit, \*publisherEdit, \*yearOfPublicationEdit, \*numberOfCopiesEdit;
13. QPushButton \*btnAdd, \*btnExit;
14. **public**:
15. addForm();
16. **public** slots:
17. **void** addRecord();
18. };
20. #endif // ADDFORM\_H
21. #include "addform.h"
22. #include "bookfile.h"
23. #include "mainform.h"
24. #include <QVBoxLayout>
25. addForm::addForm()
26. {
27. **this**->setWindowTitle("Create/Add Data About Author");
29. QVBoxLayout \*layoutV1 = **new** QVBoxLayout();
31. author = **new** QLabel("Author", **this**);
32. name = **new** QLabel("Name Of Book", **this**);
33. publisher = **new** QLabel("Publisher", **this**);
34. yearOfPublication = **new** QLabel("Year Of Publication",**this**);
35. numberOfCopies= **new** QLabel("Number Of Copies", **this**);
37. layoutV1->addWidget(author);
38. layoutV1->addWidget(name);
39. layoutV1->addWidget(publisher);
40. layoutV1->addWidget(yearOfPublication);
41. layoutV1->addWidget(numberOfCopies);
43. QVBoxLayout \*layoutV2 = **new** QVBoxLayout();
45. authorEdit = **new** QLineEdit("", **this**);
46. nameEdit = **new** QLineEdit("", **this**);
47. publisherEdit = **new** QLineEdit("", **this**);
48. yearOfPublicationEdit = **new** QLineEdit("", **this**);
49. numberOfCopiesEdit = **new** QLineEdit("", **this**);
51. layoutV2->addWidget(authorEdit);
52. layoutV2->addWidget(nameEdit);
53. layoutV2->addWidget(publisherEdit);
54. layoutV2->addWidget(yearOfPublicationEdit);
55. layoutV2->addWidget(numberOfCopiesEdit);
57. QHBoxLayout \*layoutG1 = **new** QHBoxLayout();
59. layoutG1->addLayout(layoutV1);
60. layoutG1->addLayout(layoutV2);
62. QHBoxLayout \*layoutG2 = **new** QHBoxLayout();
64. btnAdd=**new** QPushButton("Add", **this**);
65. btnExit=**new** QPushButton("Back",**this**);
67. layoutG2->addWidget(btnAdd);
68. layoutG2->addWidget(btnExit);
70. QVBoxLayout \*layout = **new** QVBoxLayout(**this**);
72. layout->addLayout(layoutG1);
73. layout->addLayout(layoutG2);
75. connect(btnAdd, SIGNAL(clicked(**bool**)),**this**,SLOT(addRecord()));
76. connect(btnExit, SIGNAL(clicked(**bool**)),**this**,SLOT(close()));
77. }
78. **void** addForm::addRecord()
79. {
80. bookFile book;
81. recType r;
82. r.author = authorEdit->text();
83. r.name = nameEdit->text();
84. r.publisher = publisherEdit->text();
85. r.yearOfPublication = yearOfPublicationEdit->text();
86. r.numberOfCopies = numberOfCopiesEdit->text();
87. authorEdit->clear();
88. nameEdit->clear();
89. publisherEdit->clear();
90. yearOfPublicationEdit->clear();
91. numberOfCopiesEdit->clear();
92. book.addRec(r);
93. }

**Код модуля “Работа с файлом”:**

1. #ifndef BOOKFILE\_H
2. #define BOOKFILE\_H
3. #include <QFile>
5. **struct** recType   // структура записи для одного абонента
6. {
7. QString author, name, publisher, yearOfPublication, numberOfCopies;
8. };
10. **class** bookFile
11. {
12. QFile \*f;    // указатель на объект класса QFile
13. **bool** k1,k2,k3,k4,ff; // ключи поиска
14. **public**:
15. **int** a[50];
16. recType r;  // буфер ввода-вывода
17. bookFile();  // конструктор
18. ~bookFile(); // деструктор
19. **bool** addRec(recType r); // добавление записи в файл
20. **bool** readRec();     // чтение записи из файда
21. **bool** findFirst(**const** recType r1, **int** p, QString number); // поиск первой
22. **bool** findNext(**const** recType r1, **int** p, QString number);
23. **int** \*findYears(**int** &size);
24. };
26. #endif // BOOKFILE\_H
27. #include "bookfile.h"
28. #include "mainform.h"
29. #include <QMessageBox>
31. bookFile::bookFile()  // конструктор
32. {
33. f = **new** QFile("book.txt"); // создаем объект-файл
34. f->open(QFile::ReadWrite); // открываем файл
35. **for** (**int** i = 0; i < 20;i++)
36. {
37. a[i] = 0;
38. }
39. }
40. bookFile::~bookFile()   // деструктор
41. {
42. f->close();        // закрываем файл
43. **delete** f;                        // освобождаем память
44. }
45. **bool** bookFile::addRec(recType r)
46. {
47. f->seek(f->size()); // переходим на конец файла
48. QDataStream out(f); // связываем с файлом поток вывода
49. out << r.author << r.name << r.publisher << r.yearOfPublication << r.numberOfCopies;// выводим данные в файл
50. **return** **true**;
51. }
52. **bool** bookFile::readRec()
53. {
54. QDataStream in(f); // связываем с файлом поток ввода
55. **if** (in.atEnd())
56. **return** **false**;
57. **else**
58. {
59. in >> r.author >> r.name >> r.publisher >> r.yearOfPublication >> r.numberOfCopies;
60. **return** **true**;
61. }
62. }
63. **bool** bookFile::findFirst(**const** recType r1, **int** i, QString number)
64. {
65. **switch** (i)
66. {
67. **case** 1:
68. {
69. k1=(r1.author == "");   // устанавливаем два ключа поиска
70. ff=**false**;   // устанавливаем ключ поиска «запись не найдена»
71. f->reset();
72. **bool** fff = readRec();
73. **while**(fff &&(!ff))
74. {
75. k3=(r1.author==r.author);  //строим еще два ключа поиска
76. **if** (!k1 && k3)
77. ff=**true**; // ключ поиска «запись найдена»
78. **else** fff=readRec();
79. }
80. **break**;
81. }
83. **case** 2:
84. {
85. ff=**false**;   // устанавливаем ключ поиска «запись не найдена»
86. f->reset();
87. **bool** fff = readRec();
88. **while**(fff &&(!ff))
89. {
90. **if** ((number.toInt() < r.numberOfCopies.toInt()))
91. ff = **true**; // ключ поиска «запись найдена»
92. **else** fff=readRec();
93. }
94. **break**;
95. }
97. **case** 3:
98. {
99. k1=(r1.publisher == "");   // устанавливаем два ключа поиска
100. k2=(r1.yearOfPublication == "");
101. ff=**false**;   // устанавливаем ключ поиска «запись не найдена»
102. f->reset();
103. **bool** fff = readRec();
104. **while**(fff &&(!ff))
105. {
106. k3=(r1.publisher==r.publisher);  //строим еще два ключа поиска
107. k4=(r1.yearOfPublication==r.yearOfPublication);
108. **if** ((!k1 && !k2 && k3 && k4)||
109. (!k1 && k2 && k3)||
110. (k1 && !k2 && k4))
111. ff=**true**; // ключ поиска «запись найдена»
112. **else** fff=readRec();
113. }
114. **break**;
115. }
116. }
117. **return** ff; // возвращаем ключ поиска
118. }
119. **bool** bookFile::findNext(**const** recType r1, **int** i, QString number)
120. {
121. ff=**false**;   // ключ поиска «запись не найдена»
122. **switch** (i)
123. {
124. **case** 1:
125. {
126. **bool** fff = readRec();
127. **while**((!ff) && fff)
128. {
129. k3 = (r1.author == r.author);//строим еще два ключа поиска
130. **if** ((!k1 && k3))
131. ff=**true**;     // ключ поиска «запись найдена»
132. **else** fff = readRec();
133. }
134. **break**;
135. }
137. **case** 2:
138. {
139. **bool** fff = readRec();
140. **while**((!ff) && fff)
141. {
142. **if** ((number.toInt() < r.numberOfCopies.toInt()) )
143. ff = **true**;     // ключ поиска «запись найдена» //что-то тут исправить
144. **else** fff = readRec();
145. }
146. **break**;
147. }
149. **case** 3:
150. {
151. **bool** fff = readRec();
152. **while**((!ff) && fff)
153. {
154. k3=(r1.publisher == r.publisher);//строим еще два ключа поиска
155. k4=(r1.yearOfPublication == r.yearOfPublication);
156. **if** (!k1 && !k2 && k3 && k4)
157. ff=**true**;     // ключ поиска «запись найдена»
158. **else** fff=readRec();
159. }
160. **break**;
161. }
162. }
163. **return** ff; // возвращаем ключ поиска
164. }
166. **int** \*bookFile::findYears(**int** &size)
167. {
168. size = 0;
169. f->reset();
170. **bool** fff = readRec();
171. **while**(fff)
172. {
173. a[size] = r.yearOfPublication.toInt();
174. fff = readRec();
175. size++;
176. }
177. **return** a;
178. }

**Код модуля “Найти авторов по количеству книг”:**

1. #ifndef FINDAUTHORSBYNUMBERSOFBOOKSFORM\_H
2. #define FINDAUTHORSBYNUMBERSOFBOOKSFORM\_H
3. #include "printform.h"
4. #include <QtGui>
5. #include <QWidget>
6. #include <QLabel>
7. #include <QLineEdit>
8. #include <QPushButton>
9. **class** findAuthorsByNumbersOfBooksForm : **public** QWidget
10. {
11. Q\_OBJECT
12. QLabel \*numberOfCopies;
13. QLineEdit \*numberOfCopiesEdit;
14. QPushButton \*btnFind, \*btnExit;
15. printForm winPrint;
16. **public**:
17. findAuthorsByNumbersOfBooksForm();        // конструктор
18. **public** slots:
19. **void** findRecs();  // метод поиска записей
20. };
22. #endif // FINDAUTHORSBYNUMBERSOFBOOKSFORM\_H
23. #include "findauthorsbynumbersofbooksform.h"
24. #include "bookfile.h"
25. #include <QVBoxLayout>
26. #include <QLabel>
27. #include <QPushButton>
28. #include <QHBoxLayout>
29. findAuthorsByNumbersOfBooksForm::findAuthorsByNumbersOfBooksForm()
30. {
31. **this**->setWindowTitle("Find All Books Of Authors");
33. QVBoxLayout \*layoutV1 = **new** QVBoxLayout();
35. numberOfCopies = **new** QLabel("Enter Number Of Books", **this**);
37. layoutV1->addWidget(numberOfCopies);
39. QVBoxLayout \*layoutV2 = **new** QVBoxLayout();
40. numberOfCopiesEdit=**new** QLineEdit("", **this**);
42. layoutV2->addWidget(numberOfCopiesEdit);
44. QHBoxLayout \*layoutG1 = **new** QHBoxLayout();
46. layoutG1->addLayout(layoutV1);
47. layoutG1->addLayout(layoutV2);
49. QHBoxLayout \*layoutG2 = **new** QHBoxLayout();
51. btnFind=**new** QPushButton("Find", **this**);
52. btnExit=**new** QPushButton("Back", **this**);
54. layoutG2->addWidget(btnFind);
55. layoutG2->addWidget(btnExit);
57. QVBoxLayout \*layout = **new** QVBoxLayout(**this**);
59. layout->addLayout(layoutG1);
60. layout->addLayout(layoutG2);
62. connect(btnFind, SIGNAL(clicked(**bool**)),
63. **this**,SLOT(findRecs()));
64. connect(btnExit, SIGNAL(clicked(**bool**)),
65. **this**,SLOT(close()));
66. }
67. **void** findAuthorsByNumbersOfBooksForm::findRecs()
68. {
69. QString number = numberOfCopiesEdit->text();
70. recType r;  // параметры поиска
71. r.numberOfCopies = numberOfCopiesEdit->text();
72. winPrint.showResults(r, 2, number);
73. numberOfCopiesEdit->clear();
74. }

**Код модуля “Найти авторов определенного издательства и определенного года”:**

1. #ifndef FINDAUTHORSBYPUBLISHINGANDYEARFORM\_H
2. #define FINDAUTHORSBYPUBLISHINGANDYEARFORM\_H
3. #include "printform.h"
4. #include <QtGui>
5. #include <QWidget>
6. #include <QLabel>
7. #include <QLineEdit>
8. #include <QPushButton>
9. **class** findAuthorsByPublishingAndYearForm : **public** QWidget
10. {
11. Q\_OBJECT
12. QLabel \*publisher, \*yearOfPublication;
13. QLineEdit \*publisherEdit, \*yearOfPublicationEdit;
14. QPushButton \*btnFind, \*btnExit;
15. printForm winPrint;
16. **public**:
17. findAuthorsByPublishingAndYearForm();        // конструктор
18. **public** slots:
19. **void** findRecs();  // метод поиска записей
20. };
22. #endif // FINDAUTHORSBYPUBLISHINGANDYEARFORM\_H
23. #include "findauthorsbypublishingandyearform.h"
24. #include "bookfile.h"
25. #include <QVBoxLayout>
26. #include <QLabel>
27. #include <QPushButton>
28. #include <QHBoxLayout>
29. findAuthorsByPublishingAndYearForm::findAuthorsByPublishingAndYearForm()
30. {
31. **this**->setWindowTitle("Find Authors By Publisher And Year Of Publication");
33. QVBoxLayout \*layoutV1 = **new** QVBoxLayout();
35. publisher = **new** QLabel(("Enter Publisher"), **this**);
36. yearOfPublication = **new** QLabel(("Enter Year Of Publication"), **this**);
38. layoutV1->addWidget(publisher);
39. layoutV1->addWidget(yearOfPublication);
41. QVBoxLayout \*layoutV2 = **new** QVBoxLayout();
43. publisherEdit = **new** QLineEdit((""), **this**);
44. yearOfPublicationEdit = **new** QLineEdit((""), **this**);
46. layoutV2->addWidget(publisherEdit);
47. layoutV2->addWidget(yearOfPublicationEdit);
49. QHBoxLayout \*layoutG1 = **new** QHBoxLayout();
51. layoutG1->addLayout(layoutV1);
52. layoutG1->addLayout(layoutV2);
54. QHBoxLayout \*layoutG2 = **new** QHBoxLayout();
56. btnFind=**new** QPushButton(("Find"), **this**);
57. btnExit=**new** QPushButton(("Back"), **this**);
59. layoutG2->addWidget(btnFind);
60. layoutG2->addWidget(btnExit);
62. QVBoxLayout \*layout = **new** QVBoxLayout(**this**);
64. layout->addLayout(layoutG1);
65. layout->addLayout(layoutG2);
67. resize(450,100);
69. connect(btnFind, SIGNAL(clicked(**bool**)),
70. **this**,SLOT(findRecs()));
71. connect(btnExit, SIGNAL(clicked(**bool**)),
72. **this**,SLOT(close()));
73. }
74. **void** findAuthorsByPublishingAndYearForm::findRecs()
75. {
76. QString number = "";
77. recType r;  // параметры поиска
78. r.publisher = publisherEdit->text();
79. r.yearOfPublication = yearOfPublicationEdit->text();
80. winPrint.showResults(r, 3, number);
81. publisherEdit->clear();
82. yearOfPublicationEdit->clear();
83. }

**Код модуля “Найти автора”:**

1. #ifndef FINDAUTHORSFORM\_H
2. #define FINDAUTHORSFORM\_H
3. #include "printform.h"
4. #include <QtGui>
5. #include <QWidget>
6. #include <QLabel>
7. #include <QLineEdit>
8. #include <QPushButton>
9. **class** findAuthorsForm : **public** QWidget
10. {
11. Q\_OBJECT
12. QLabel \*author;
13. QLineEdit \*authorEdit;
14. QPushButton \*btnFind, \*btnExit;
15. printForm winPrint;
16. **public**:
17. findAuthorsForm();        // конструктор
18. **public** slots:
19. **void** findRecs();  // метод поиска записей
20. };
22. #endif // FINDAUTHORSFORM\_H
23. #include "findauthorsform.h"
24. #include "bookfile.h"
25. #include <QVBoxLayout>
26. #include <QLabel>
27. #include <QPushButton>
28. #include <QHBoxLayout>
29. findAuthorsForm::findAuthorsForm()
30. {
31. **this**->setWindowTitle("Find All Books Of Authors");
33. QVBoxLayout \*layoutV1 = **new** QVBoxLayout();
35. author = **new** QLabel("Enter The Author Of The Book", **this**);
37. layoutV1->addWidget(author);
39. QVBoxLayout \*layoutV2 = **new** QVBoxLayout();
40. authorEdit=**new** QLineEdit("", **this**);
42. layoutV2->addWidget(authorEdit);
44. QHBoxLayout \*layoutG1 = **new** QHBoxLayout();
46. layoutG1->addLayout(layoutV1);
47. layoutG1->addLayout(layoutV2);
49. QHBoxLayout \*layoutG2 = **new** QHBoxLayout();
51. btnFind=**new** QPushButton("Find", **this**);
52. btnExit=**new** QPushButton("Back", **this**);
54. layoutG2->addWidget(btnFind);
55. layoutG2->addWidget(btnExit);
57. QVBoxLayout \*layout = **new** QVBoxLayout(**this**);
59. layout->addLayout(layoutG1);
60. layout->addLayout(layoutG2);
62. connect(btnFind, SIGNAL(clicked(**bool**)),
63. **this**,SLOT(findRecs()));
64. connect(btnExit, SIGNAL(clicked(**bool**)),
65. **this**,SLOT(close()));
66. }
67. **void** findAuthorsForm::findRecs()
68. {
69. QString number = "";
70. recType r;  // параметры поиска
71. r.author = authorEdit->text();
72. winPrint.showResults(r, 1, number);
73. authorEdit->clear();
74. }

**Код модуля “Вывод результата”:**

1. #ifndef PRINTFORM\_H
2. #define PRINTFORM\_H
3. #include "bookfile.h"
4. #include <QtGui>
5. #include <QTableWidget>
6. #include <QPushButton>
7. **class** printForm : **public** QWidget
8. {
9. QTableWidget \*table;  // таблица
10. QPushButton \*btnExit;
11. **void** showRow(**int** i,recType r);// вывод строки таблицы
12. **public**:
13. printForm();    // конструктор
14. **void** showResults(recType r1, **int** p, QString number); // показать результаты поиска
15. };
17. #endif // PRINTFORM\_H
18. #include "printform.h"
19. #include "mainform.h"
20. #include <QMessageBox>
21. #include <QHBoxLayout>
22. printForm::printForm()
23. {
24. **this**->setWindowTitle("Result");
25. QStringList strlist; // объект Список строк
26. strlist << "Author" << "Name Of Book" << "Publisher" << "Number Of Copies" << "Year Of Publication"; // записываем строки заголовка
27. table = **new** QTableWidget(20,5,**this**); // создаем таблицу
28. table->setHorizontalHeaderLabels(strlist);// включаем
29. //  заголовок
30. QHBoxLayout \*layoutG2 = **new** QHBoxLayout();
32. btnExit=**new** QPushButton("Back", **this**);
33. layoutG2->addWidget(btnExit);
35. QVBoxLayout \*layout = **new** QVBoxLayout(**this**);
37. layout->addWidget(table);
38. layout->addLayout(layoutG2);
40. connect(btnExit, SIGNAL(clicked(**bool**)),**this**,SLOT(close()));
41. }
42. **void** printForm::showRow(**int** i,recType r)
43. {
44. QTableWidgetItem \*item; // элемент таблицы
45. item = **new** QTableWidgetItem(); // создаем элемент
46. item->setFlags(Qt::NoItemFlags);//запрещаем
47. // выделение
48. item->setText(r.author);   // записываем текст
49. table->setItem(i,0,item);// привязываем элемент
50. // к таблице
51. item = **new** QTableWidgetItem();// создаем элемент
52. item->setFlags(Qt::NoItemFlags); //запрещаем
53. // выделение
54. item->setText(r.name);
55. table->setItem(i,1,item); // привязываем элемент
57. item = **new** QTableWidgetItem();// создаем элемент
58. item->setFlags(Qt::NoItemFlags); //запрещаем
59. // выделение
60. item->setText(r.publisher); // записываем текст
61. table->setItem(i,2,item); // привязываем элемент
63. item = **new** QTableWidgetItem();// создаем элемент
64. item->setFlags(Qt::NoItemFlags); //запрещаем
66. item->setText(r.numberOfCopies); // записываем текст
67. table->setItem(i,3,item); // привязываем элемент
69. item = **new** QTableWidgetItem();// создаем элемент
70. item->setFlags(Qt::NoItemFlags); //запрещаем
72. item->setText(r.yearOfPublication); // записываем текст
73. table->setItem(i,4,item); // привязываем элемент
75. item = **new** QTableWidgetItem();// создаем элемент
76. item->setFlags(Qt::NoItemFlags); //запрещаем
77. }
79. **void** printForm::showResults(recType r1, **int** p, QString number)
80. {
81. bookFile book;
82. **if** (!book.findFirst(r1, p, number))
83. {   // если данные не найдены, то создаем сообщение
84. QMessageBox msg(QMessageBox::Critical, "No Data", "No Data Found", QMessageBox::Ok, 0);
85. msg.exec();
86. }
87. **else**
88. {   // иначе - выводим результаты  по строкам
89. showRow(0, book.r);
90. **int** i = 0;
91. **while** (book.findNext(r1, p, number))
92. showRow(++i, book.r);
93. table->setRowCount(i+1);
94. resize(550,400);
95. show();
96. }
97. }

**Код модуля “Диаграмма распределения книг”:**

1. #ifndef WIDGET\_H
2. #define WIDGET\_H
3. #include <QVBoxLayout>
4. #include <QPushButton>
5. #include "qcustomplot.h"
6. #include "bookfile.h"
8. **class** Widget : **public** QWidget
9. {
10. Q\_OBJECT
12. **public**:
13. **explicit** Widget(QWidget \*parent = 0);
15. **private**:
16. **int** key;
17. QCustomPlot \*customPlot;
18. QVBoxLayout \*vbox;
19. QPushButton \*res, \*btnExit;
20. QCPBars \*fossil;
22. **private** slots:
23. **void** rndres();
24. };
26. #endif // WIDGET\_H
27. #include "widget.h"
28. #include "qcustomplot.h"
29. #include "bookfile.h"
31. Widget::Widget(QWidget \*parent) :
32. QWidget(parent)
33. {
34. resize(600,400);
35. setWindowTitle(QString::fromUtf8("Diagram Of Distribution Of Books"));
36. customPlot = **new** QCustomPlot(**this**);
37. vbox = **new** QVBoxLayout(**this**);

40. vbox->addWidget(customPlot);
41. res = **new** QPushButton(QString::fromUtf8("Show Or Update Diagram"),**this**);
42. btnExit = **new** QPushButton("Close", **this**);
43. vbox->addWidget(res);
44. vbox->addWidget(btnExit);
45. setLayout(vbox);
47. // Сигналы
48. connect(res,SIGNAL(clicked()),**this**,SLOT(rndres()));
49. connect(btnExit, SIGNAL(clicked()), **this** ,SLOT(close()));
50. }
52. **void** Widget::rndres()
53. {
55. fossil = **new** QCPBars(customPlot->xAxis, customPlot->yAxis);
57. customPlot->addPlottable(fossil);
58. // Установки цвета:
59. QPen pen;
60. pen.setWidthF(2);//Толщина контура столбца
61. fossil->setName(QString::fromUtf8("Books")); // Легенда
62. pen.setColor(QColor(50, 50, 100));// Цвет контура столбца
63. fossil->setPen(pen);
64. // Цвет самого столбца, четвертый параметр - прозрачность
65. fossil->setBrush(QColor(50, 50, 250, 70));
67. bookFile book;
68. **int** size = 0;
69. book.findYears(size);
70. **int** \*mas1 = **new** **int**[size];
71. **int** \*mas2 = **new** **int**[size];
73. **for** (**int** i = 0; i < size; i++)
74. {
75. mas2[i] = 1;
76. mas1[i] = 0;
77. }
79. **for** (**int** i = 0; i < size; i++)
80. {
81. mas1[i] = book.a[i];
82. }
84. **for** (**int** i = 0; i < size - 1; i++)
85. {
86. **for** (**int** j = (size - 1); j > i; j--)
87. {
88. **if** (mas1[j - 1] > mas1[j])
89. {
90. **int** temp = mas1[j - 1];
91. mas1[j - 1] = mas1[j];
92. mas1[j] = temp;
93. }
94. }
95. }
97. **for** (**int** i = 0; i < size - 1; i++)
98. {
99. **for** (**int** k = i + 1; k < size; k++)
100. {
101. **if** ((mas1[i] == mas1[k]) && (mas1[i] != 0) && (mas1[k] != 0))
102. {
103. mas2[i]++;
104. mas1[k] = 0;
105. mas2[k]--;
106. }
107. }
108. }
110. // Установки значений оси X:
111. **int** sizeXAxis = size;
112. **for** (**int** i = 0; i < size; i++)
113. {
114. **if** ((mas2[i] == 0) || (mas1[i] == 0))
115. {
116. sizeXAxis--;
117. }
118. }
119. QVector<**double**> ticks;
120. QVector<QString> labels;
121. **for** (**int** k = 0; k < sizeXAxis; k++)
122. {
123. ticks << k;
124. }
125. QString s;
126. **for** (**int** i = 0; i < size;i++)
127. {
128. **if** (mas1[i] != 0)
129. {
130. s = QString::number(mas1[i]);
131. labels << s;
132. }
133. }
134. customPlot->xAxis->setAutoTicks(**false**);
135. customPlot->xAxis->setAutoTickLabels(**false**);
136. customPlot->xAxis->setTickVector(ticks);
137. customPlot->xAxis->setTickVectorLabels(labels);
138. customPlot->xAxis->setSubTickCount(0);
139. customPlot->xAxis->setTickLength(0, 4);
140. customPlot->xAxis->grid()->setVisible(**true**);
141. customPlot->xAxis->setRange(0, sizeXAxis);
142. customPlot->xAxis->setPadding(5);
143. customPlot->xAxis->setLabel(QString::fromUtf8("Year Of Publication"));
145. // Установки значений оси Y:
146. **int** heightMax = 0;
147. QVector<**double**> fossilData;
148. **for** (**int** i = 0; i < size; i++)
149. {
150. **if** ((mas2[i] != 0) && (mas1[i] != 0))
151. {
152. fossilData << mas2[i];
153. **if**(mas2[i] > heightMax)
154. {
155. heightMax = mas2[i]+1;
156. }
157. }
158. }
159. fossil->setData(ticks, fossilData);
160. customPlot->yAxis->setRange(0, heightMax);
161. customPlot->yAxis->setPadding(5);
162. customPlot->yAxis->setLabel(QString::fromUtf8("Number Of Books"));
163. customPlot->yAxis->grid()->setSubGridVisible(**true**);
164. QPen gridPen;
165. gridPen.setStyle(Qt::SolidLine);
166. gridPen.setColor(QColor(0, 0, 0, 25));
167. customPlot->yAxis->grid()->setPen(gridPen);
168. gridPen.setStyle(Qt::DotLine);
169. customPlot->yAxis->grid()->setSubGridPen(gridPen);
171. // Легенда:
172. customPlot->legend->setVisible(**true**);
173. customPlot->axisRect()->insetLayout()->setInsetAlignment(0, Qt::AlignTop|Qt::AlignHCenter);
174. customPlot->legend->setBrush(QColor(255, 255, 255, 200));
175. QPen legendPen;
176. legendPen.setColor(QColor(130, 130, 130, 200));
177. customPlot->legend->setBorderPen(legendPen);
178. QFont legendFont = font();
179. legendFont.setPointSize(10);
180. customPlot->legend->setFont(legendFont);
181. customPlot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom);
182. customPlot->replot();
184. // Сброс всех установок графика:
185. customPlot->removePlottable(fossil);
186. **delete** [] mas1;
187. **delete** [] mas2;
188. }

Диаграмма объектов программы ‘Электронная картотека библиотеки’ представлена на рисунке 1.1

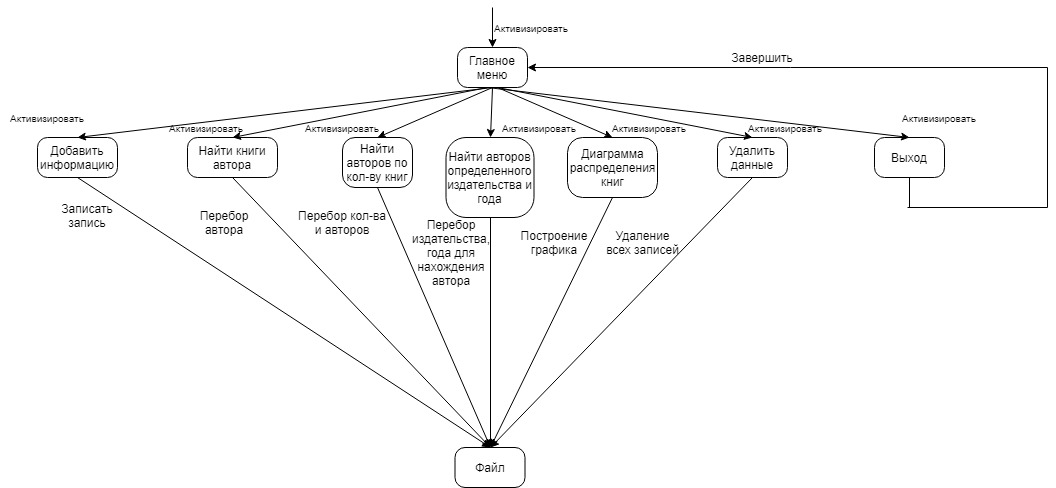


Рисунок 1.1 Диаграмма объектов

Диаграмма классов ‘Электронная картотека библиотеки’ представлена на рисунке 1.2

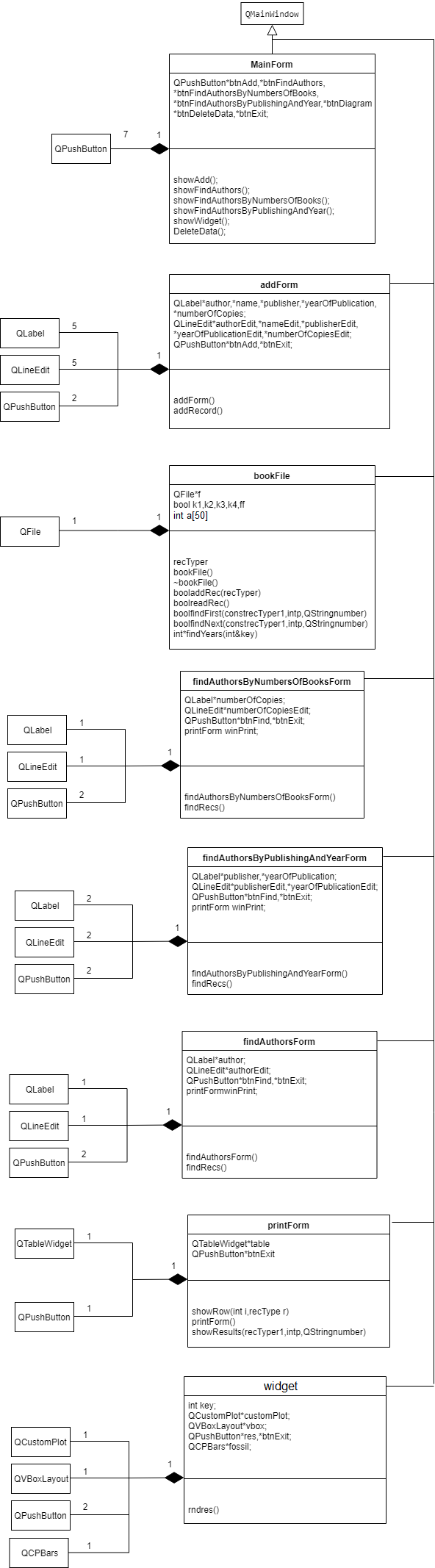


Рисунок 1.2 Диаграмма классов

Формы приложения представлены на рисунке 1.3 – 1.8

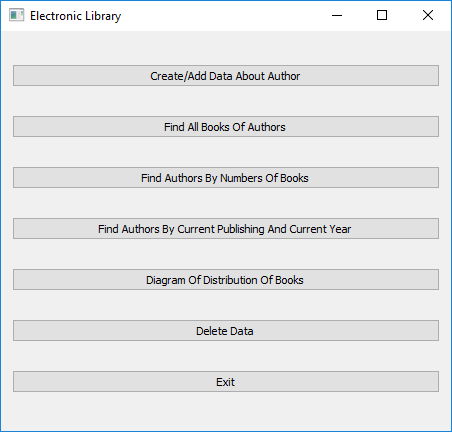


Рисунок 1.3 Внешний вид главного окна приложения

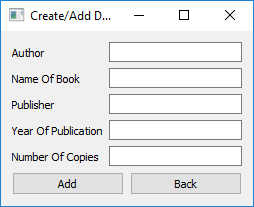


Рисунок 1.4 Внешний вид формы ‘Создать/Добавить информацию об авторе’

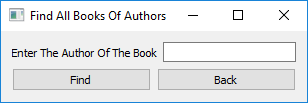


Рисунок 1.5 Внешний вид формы ‘Найти все книги автора’

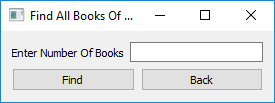


Рисунок 1.6 Внешний вид формы ‘Найти авторов по кол-ву книг’

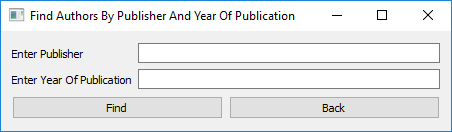


Рисунок 1.7 Внешний вид формы ‘Найти все книги авторов данного издательства и данного года’

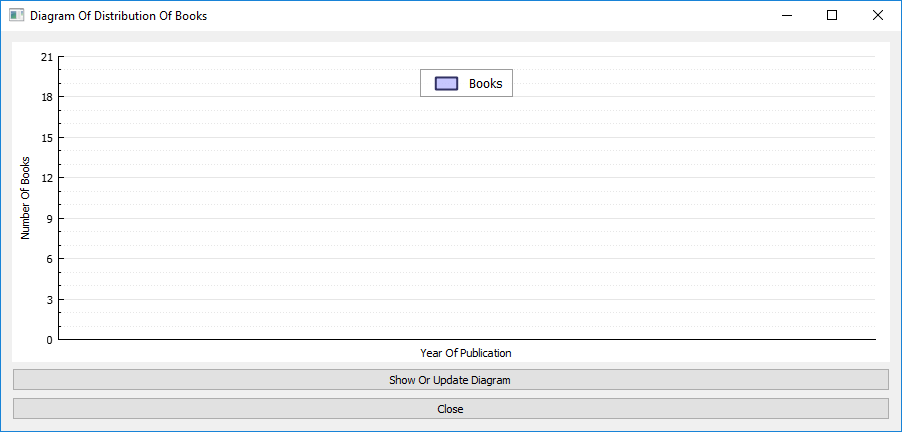


Рисунок 1.8 Внешний вид формы ‘Диаграмма распределения книг’

Диаграмма состояний интерфейса приложения программы ‘Электронная картотека библиотеки’ представлена на рисунке 1.9 – 2.0

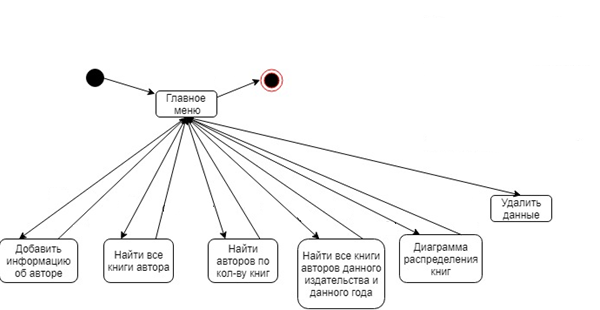


Рисунок 1.9 Диаграмма состояний интерфейса главного окна приложения

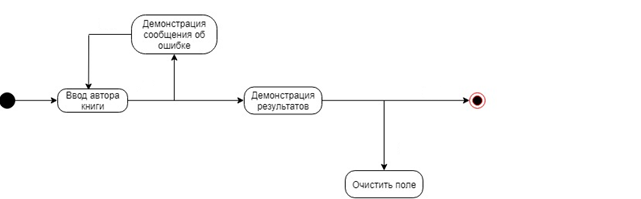


Рисунок 2.0 Диаграмма состояний интерфейса окна ‘Найти все книги автора’ приложения

Диаграмма последовательностей программы ‘Электронная картотека библиотеки’ формы ‘Создать/Добавить информацию об авторе’ представлена на рисунке 2.1

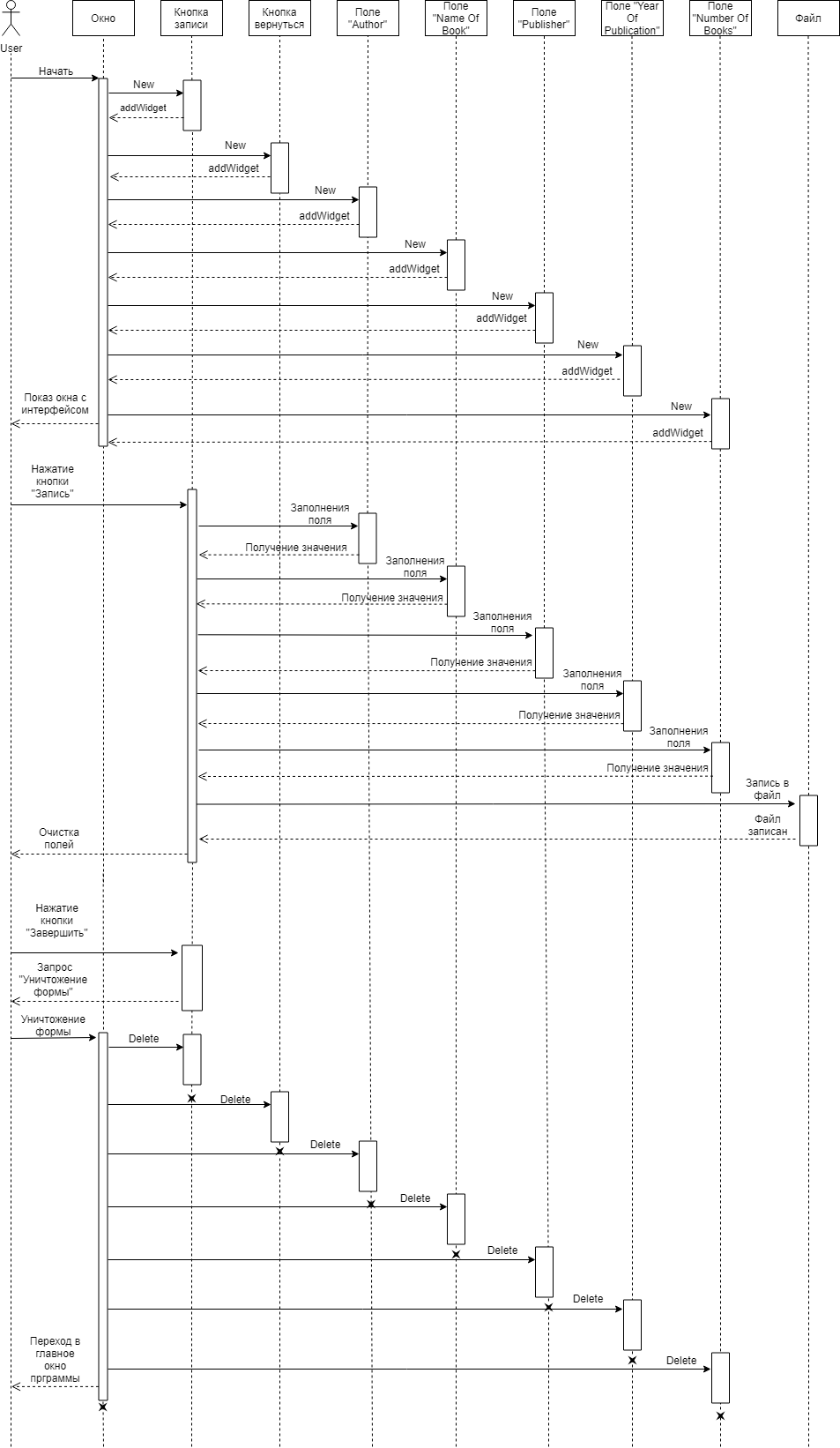


Рисунок 2.1 Диаграмма последовательностей

**Вывод:** Сделалпрограмму при помощи среды QtCreator, которая выводит на экран программу “Электронная картотека библиотеки”. “Электронная картотека библиотеки” позволяет добавлять, обрабатывать данные об авторе.