

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ОДЕССКИЙ НАЦИОНАЛЬНЫЙ МОРСКОЙ УНИВЕРСИТЕТ

КАФЕДРА "ТЕХНИЧЕСКАЯ КИБЕРНЕТИКА"

КУРСОВАЯ РАБОТА

по дисциплине «Организация баз данных»
на тему: «Библиотека. Администрация библиотеки»

Выполнил:
студент КСФ
3 курс 2 группа
Хмельницкий.Б.М

Руководитель:
Ст. преподаватель
Рублев И.С.

Одесса 2018

ОГЛАВЛЕНИЕ

1. ЗАДАЧА.....	3
2. ПОСТРОЕНИЕ ЛОГИЧЕСКОЙ И ФИЗИЧЕСКОЙ МОДЕЛИ БД.....	5
3. РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ.....	7
ВЫВОД.....	11
СПИСОК ЛИТЕРАТУРЫ.....	12
ПРИЛОЖЕНИЕ.....	13

1. ЗАДАЧА

1. Построить логическую и физическую модели БД:

Логическая модель должна учитывать максимально возможное количество требований предметной области. При необходимости можно вводить «суррогатные ключи» в проектируемые сущности. В этом случае должны быть определены соответствующие им альтернативные ключи.

- * При разработке логической модели должны быть правильно определены основные сущности, между ними установлены связи, отвечающие требованиям на добавление, изменение и удаление данных.
- * Те требования предметной области, которые не удастся выполнить при построении логической модели БД, должны быть учтены путем ограничений ссылочной целостности таблиц БД, с помощью триггеров или хранимых процедур, либо при создании клиентского приложения.

Физическая модель должна быть построена для СУБД Interbase, причем схема модели БД должны удовлетворять условиям не меньше чем НБКФ.

ОПИСАНИЕ ЗАДАЧИ:

Пусть требуется разработать информационную систему для автоматизации учета получения и выдачи книг в библиотеке. Система должна предусматривать режимы ведения системного каталога, отражающего перечень областей знаний, по которым имеются книги в библиотеке. Внутри библиотеки области знаний в систематическом каталоге могут иметь уникальный внутренний номер и полное наименование. Каждая книга может содержать сведения из нескольких областей знаний. Каждая книга в библиотеке может присутствовать в нескольких экземплярах.

Каждая книга, хранящаяся в библиотеке, характеризуется следующими параметрами:

- уникальный шифр;
- название;
- фамилии авторов (могут отсутствовать);
- место издания (город);
- издательство;
- год издания;
- количество страниц;
- стоимость книги;
- количество экземпляров книги в библиотеке.

Книги могут иметь одинаковые названия, но они различаются по своему уникальному шифру (ISBN).

В библиотеке ведется картотека читателей.

На каждого читателя в картотеку заносятся следующие сведения:

- фамилия, имя, отчество;
- домашний адрес;
- телефон (будем считать, что у нас два телефона — рабочий и домашний);
- дата рождения.

Каждому читателю присваивается уникальный номер читательского билета.

Каждый читатель может одновременно держать на руках не более 5 книг. Читатель не должен одновременно держать более одного экземпляра книги одного названия.

Каждая книга в библиотеке может присутствовать в нескольких экземплярах.

Каждый экземпляр имеет следующие характеристики:

- уникальный инвентарный номер;

- шифр книги, который совпадает с уникальным шифром из описания книг;
- место размещения в библиотеке.

В случае выдачи экземпляра книги читателю в библиотеке хранится специальный вкладыш, в котором должны быть записаны следующие сведения:

- номер билета читателя, который взял книгу;
- дата выдачи книги;
- дата возврата.

Предусмотреть следующие ограничения на информацию в системе:

1. Книга может не иметь ни одного автора.
2. В библиотеке должны быть записаны читатели не моложе 17 лет.
3. В библиотеке присутствуют книги, изданные начиная с 1960 по текущий год.
4. Каждый читатель может держать на руках не более 5 книг.
5. Каждый читатель при регистрации в библиотеке должен дать телефон для связи: он может быть рабочим или домашним.
6. Каждая область знаний может содержать ссылки на множество книг, но каждая книга может относиться к различным областям знаний.

При работе с системой библиотекарь отдела комплектации должен решать следующие задачи:

1. Принимать новые книги и регистрировать их в библиотеке.
2. Относить книги к одной или к нескольким областям знаний.
3. Проводить инвентаризацию книг, то есть назначение новых инвентарных номеров вновь принятым книгам, и, помещая их на полки библиотеки, запоминать место размещения каждого экземпляра.
4. Проводить дополнительную инвентаризацию, если поступило несколько экземпляров книги, которая уже есть в библиотеке, при этом информация о книге в предметный каталог не вносится, а каждому новому экземпляру присваивается новый инвентарный номер и для него определяется место на полке библиотеки.

1. На основании физической модели создать соответствующие таблицы БД «library».

Для всех таблиц создать необходимые триггера и одну хранимую процедуры (списание книг заданного ISBN).

2. В Qt creator создать программу-клиент для «library»

Программа должна иметь следующие формы:

Главная форма - для контроля выданных и выдачи новых книг Читателю.

Позволяет определять должников библиотеки (за текущий период) .

Вспомогательные формы:

Форма списка книг, пользующихся наибольшим спросом у читателей (по данным за 6 мес; сортировка по возрастанию популярности)

Форма - отчет

Акт списания книг заданного ISBN.

3. Использовать транзакции (блокирование таблиц) для многопользовательской работы с БД.

Результатом работы являются:

Файл и глава в отчете о созданных логической и физической моделях (сущности, их свойства, ключи; связи, их тип, ссылочная целостность)

Файлы базы данных, заполненные информацией. Описание и текст созданных триггеров, индексов и хранимых процедур БД.

Файлы программы-клиента. Распечатка текста программ и примеров - отчетов разработанных форм.

2. ПОСТРОЕНИЕ ЛОГИЧЕСКОЙ И ФИЗИЧЕСКОЙ МОДЕЛИ БД

На основании представленной задачи, мной были выделены следующие сущности и их атрибуты:

Сущность – это класс однотипных объектов, информация о которых должна быть учтена в модели.

Атрибуты сущности – это характеристика или свойство сущности, имеющая имя, уникальное для данной сущности.

При построении своей логической модели, я выделил следующие сущности: description, books, ticket, readers.

Атрибутами сущности, то есть особыми полями, которые отображают основную логику связей, так называемые первичные ключи. Помогают осуществлять точные операции над базами и помогают пользователю избежать проблем в связи заполнением и дальнейшей эксплуатацией созданной базы. В качестве примера возьмем таблицу «books», у которой есть 4 поля, одно из которых является первичным ключом – pinv.

Связи – это некоторая ассоциация между двумя сущностями. Одна сущность может быть связана с другой или сама с собой. Связи позволяют по экземпляру одной сущности находить соответствующие экземпляры другой, связанные с ней. Например, связи могут выражаться следующими фразами – экзаменов много, а экзаменационная ведомость одна, то есть перспектива связи **один ко многим**.

В сущностях базы присутствуют следующие атрибуты:

Book:

ISBN- уникальный шифр книги;
book_title- название книги;
city_of_publication- город публикации;
number_of_pages- количество страниц;
year_of_publication- год издания;
price- начальная стоимость;
written_off- флаг списания книг.

Books_of_reader:

id_reader- идентификатор читателя;
date_of_issue- дата получения книги;
date_of_return- дата возврата книги;
is_buy- флаг покупки книги;
max_number_of_days- максимальное число дней для пользования;
ISBN- уникальный шифр книги;
inventory_number- идентификатор экземпляра книги.

Copies_of_books:

inventory_number- идентификатор экземпляра книги;
ISBN- уникальный шифр книги;
date_of_receiving- дата получения книги;
room- комната;
shelf- полка;
place- позиция.

Money:

inventory_number- идентификатор экземпляра книги;
compensation_money- деньги уплаченные вместо книги;
fine_money- деньги уплаченные за простроченное возвращение.

Readers:

id_reader- идентификатор читателя;
mobile_phone- мобильный номер телефона;
mobile_phone- домашний номер телефона;
date_of_birth- дата рождения;
name- имя;
surname- фамилия;
patronymic- отчество.

Authors_of_the_book:

ISBN- уникальный шифр книги;
id_author- идентификатор автора.

Authors:

id_author- отчество;
name- имя;
surname- фамилия;
patronymic- отчество.

Genres_of_books:

ISBN- уникальный шифр книги;
id_genre- идентификатор жанра.

Thems:

id_genre- идентификатор жанра;
genre- название жанра.

Логическая модель структуры базы данных приведена на рисунке 1.

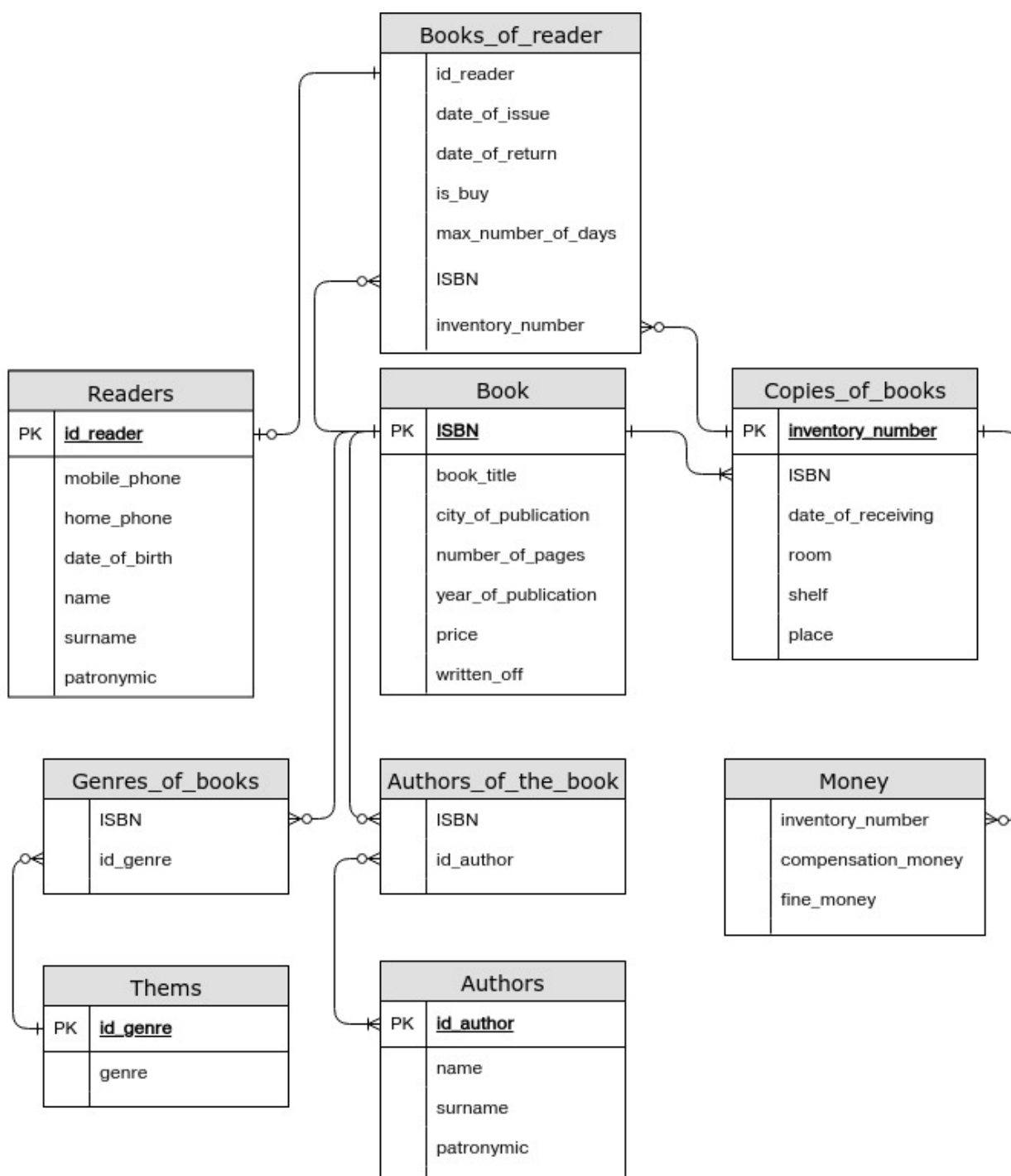


Рисунок 1 — логическая структура базы данных «library»

3. РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

Клиентское приложение создавалось мною в среде разработки Qt Creator 4.4.1 на языке программирования C++. Данное клиентское приложение разработано с помощью технологии QSqlDatabase и имеет 6 форм.

Для защиты от несанкционированного доступа была создана форма авторизации рисунок 2.

Главное окно приложения поделено на три вкладки для оперирования информацией о читателях рисунок 3, просмотра популярности книг и возможности списывания рисунок 4, а также вкладка с отображением текущего статуса каждого экземпляра книги рисунок 5. Вывод списка читателей доступен по следующим категориям: все читатели, должники, без задолженностей и читатели, что не держат на руках книг. Для читателя можно выписать книгу: рисунок 6, а также принять книгу: рисунок 7. В случае если сдача книги просрочена, то открывается форма для внесения оплаты рисунок 8.

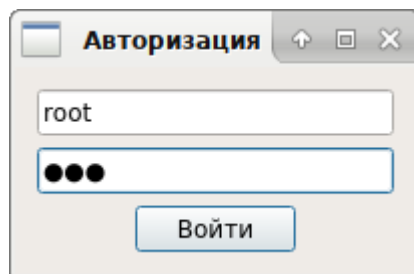
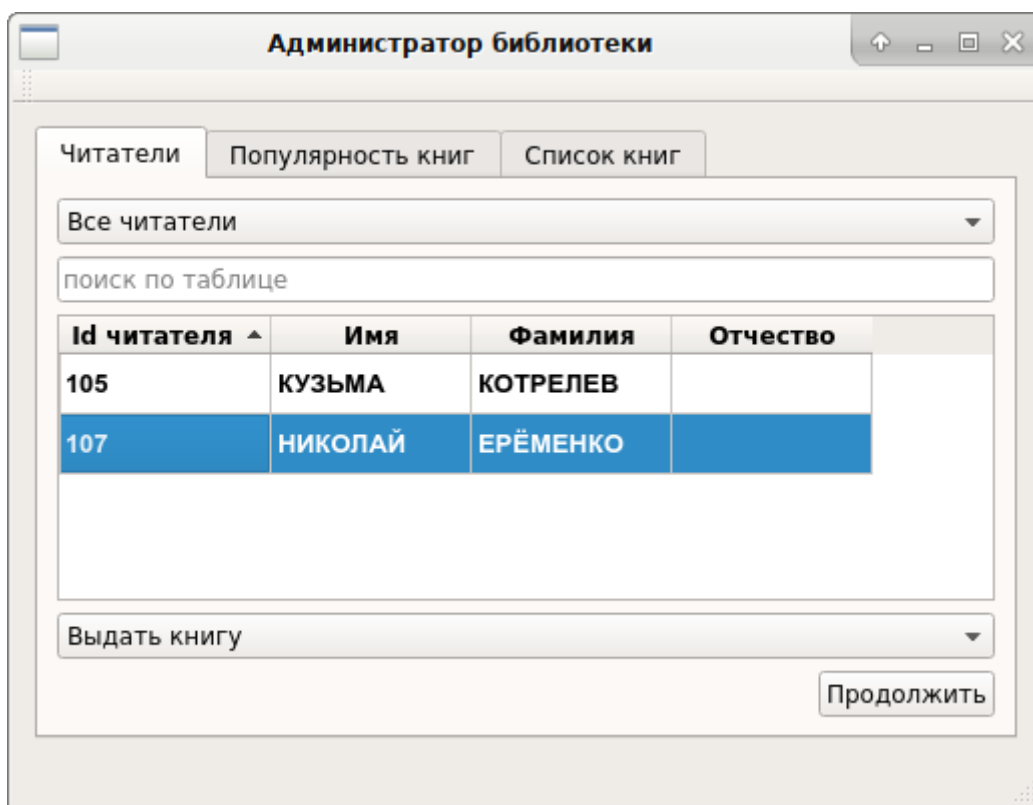


Рисунок 2 — Форма авторизации



Id читателя	Имя	Фамилия	Отчество
105	КУЗЬМА	КОТРЕЛЕВ	
107	НИКОЛАЙ	ЕРЁМЕНКО	

Рисунок 3 — Форма управления читателями

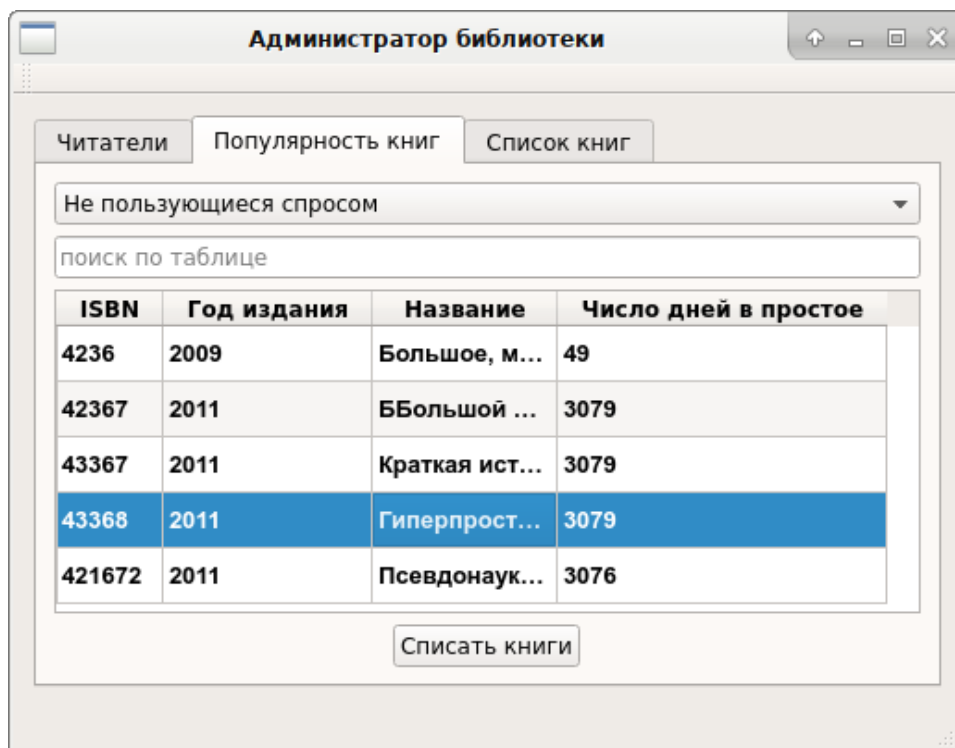


Рисунок 4 — Популярность книг

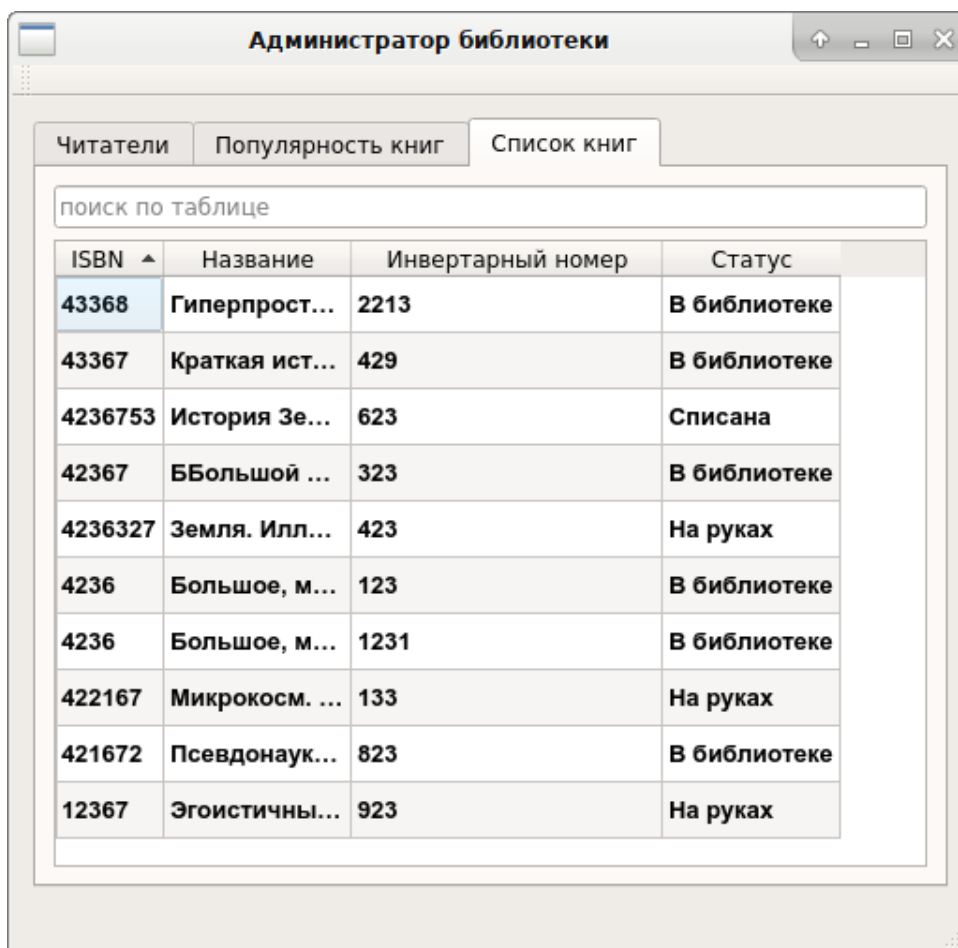


Рисунок 5 — Текущий статус для всех экземпляров книг

Выдача книги

поиск по таблице

ISBN ^	Инвентарный номер	Название
4236	123	Большое, м...
42367	323	ББольшой ...
43367	429	Краткая ист...
421672	823	Псевдонаук...
4236	1231	Большое, м...
43368	2213	Гиперпрост...

25

Выдать книгу

Рисунок 6 — Форма выдачи читателю новой книги

Возврат книги

поиск по таблице

ISBN ^	Инвентарный номер	Название	Число оставшихся дней
4236327	423	Земля. Илл...	70
422167	133	Микрокосм. ...	-222
43368	2213	Гиперпрост...	25

Принять книгу

Рисунок 7 — Форма возврата книги

Оплата штрафа

Сдача книги просрочена на 222 дней

Оплата за штраф

Оплатить

Рисунок 8 — Форма оплаты штрафа

ВЫВОД

В курсовом проекте мною было проведено проектирование и создание базы данных, а также клиентского приложения в среде разработки Qt Creator 4.4.1 в соответствии с заданием проекта.

При этом в процессе работы была создана база данных предназначенная для формирования и хранения данных по книгам, хранящимся в библиотеке. Кроме того, база данных позволяет хранить информацию количестве книг, их расположению на полке, а количество страниц в книге.

Созданное к базе данных клиентское приложение позволяет быстро выдавать и принимать у читателей книги, а также списывать старые книги.

СПИСОК ЛИТЕРАТУРЫ

1. Страуструп Б. Язык программирования C++, 4-е видання— М: Бином, 2015, – 1136с
2. Стефан Р. Девіс. C++ For Dummies, 6-е видання – Willey Publishing Inc., 2009. – 336с.
3. Герберт Шилд. Самоучитель C++, 3-е видання. Пер. з англ. – Петербург : БХВ, 2003. – 688 с.
4. Ларри Ульман MYSQL Руководство по изучению языка. —М.: МГОУ, 2006, — 109с.

ПРИЛОЖЕНИЕ

Листинг главной формы «MainWindow»:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    initialize();
    displayRequest(); //сделать авторизацию
}
MainWindow::~MainWindow()
{
    delete comboReaderGroup1;
    delete lineSearch1;
    delete tableOfReaders1;
    delete comboAction1;
    delete buttonAgree1;
    delete comboBookPopularity2;
    delete lineSearch2;
    delete tableOfBooks2;
    delete buttonWriteOff2;
    delete lineSearch3;
    delete tableOfBooks3;
    delete tabLayout1;
    delete tabLayout2;
    delete tabLayout3;
    delete tab1;
    delete tab2;
    delete tab3;
    delete ui;
}
void MainWindow::displayRequest()
{
    Form_Login *formLogin = new Form_Login(this); //авторизация
    connect(formLogin, SIGNAL(finished(int)) ,this,SLOT(formLoginDestroy(int)
));
    //this->setVisible(false);
    formLogin->show();
    /*
    bool t = BD::getInstance()->createConnect("root","123");
    qDebug()<<t;
    //BD::getInstance()->make("insert into worker(name,dept) values('Ritu23',
'Accounting')");
    QSqlQuery *query = BD::getInstance()->make("select * from book");
    if(query == NULL)
        return;
    while(query->next())
    {
        QString type = query->value(1).toString();
        qDebug()<<type;
    }*/
}
void MainWindow::initialize()
{
    comboReaderGroup1=0;
    lineSearch1=0;
    tableOfReaders1=0;
    comboAction1=0;
    buttonAgree1=0;
```

```

        comboBookPopularity2=0;
        lineSearch2=0;
        tableOfBooks2=0;
        buttonWriteOff2=0;
        lineSearch3=0;
        tableOfBooks3=0;
        tabLayout1=0;
        tabLayout2=0;
        tabLayout3=0;
        tab1=0;
        tab2=0;
        tab3=0;
    }
    void MainWindow::buid()
    {
        ui->tabWidget->clear();//удалить все вкладки
        createTab1();
        createTab2();
        createTab3();
        ui->tabWidget->addTab(tab1,"Читатели");
        ui->tabWidget->addTab(tab2,"Популярность книг");
        ui->tabWidget->addTab(tab3,"Список книг");
        ui->tabWidget->setCurrentWidget(tab1);
    }
    void MainWindow::createTab1()
    {
        //tab1
        tab1 = new QWidget();
        tabLayout1 = new QVBoxLayout();
        comboReaderGroup1 = new QComboBox();
        comboReaderGroup1->addItem("Все читатели");
        comboReaderGroup1->addItem("Должники");
        comboReaderGroup1->addItem("Без задолженностей");
        comboReaderGroup1->addItem("Читатели без книг");
        connect(comboReaderGroup1, SIGNAL(currentIndexChanged(int)), this,
        SLOT(Reader_IndexChanged(int)) );
        lineSearch1 = new QLineEdit();
        lineSearch1->setPlaceholderText("поиск по таблице");
        connect(lineSearch1, SIGNAL(textEdited(QString)), this,
        SLOT(edited_Search_1(QString)) );
        tableOfReaders1 = new TableViewOfBase();
        connect(tableOfReaders1, SIGNAL(clicked(QModelIndex)),
        SLOT(slotClikTable_1(const QModelIndex)) );
        comboAction1 = new QComboBox();
        comboAction1->addItem("Выдать книгу");
        comboAction1->addItem("Принять книгу");
        comboAction1->addItem("Возместить утрату");
        comboAction1->addItem("Сведения о взятых книгах");
        connect(comboAction1, SIGNAL(currentIndexChanged(int)), this,
        SLOT(Action_IndexChanged(int)) );
        buttonAgree1 = new QPushButton("Продолжить");
        connect(buttonAgree1, SIGNAL(released()), SLOT(buttononAgree1_Released())
        );
        tabLayout1->addWidget(comboReaderGroup1);
        tabLayout1->addWidget(lineSearch1);
        tabLayout1->addWidget(tableOfReaders1);
        tabLayout1->addWidget(comboAction1);
        tabLayout1->addWidget(buttonAgree1);
        tabLayout1->setAlignment(buttonAgree1,Qt::AlignRight);
        tab1->setLayout(tabLayout1);
        activateTab_1(false);
        updatetableOfReaders(comboReaderGroup1->currentIndex(),lineSearch1-
        >text());
    }

```

```

void MainWindow::createTab2()
{
    //tab2
    tab2 = new QWidget();
    tabLayout2 = new QVBoxLayout();
    comboBookPopularity2 = new QComboBox();
    comboBookPopularity2->addItem("Нииболее популярные");
    comboBookPopularity2->addItem("Не пользующиеся спросом");
    connect(comboBookPopularity2, SIGNAL(currentIndexChanged(int)), this,
    SLOT(BookPopularity_IndexChanged(int)) );
    lineSearch2 = new QLineEdit();
    lineSearch2->setPlaceholderText("поиск по таблице");
    connect(lineSearch2, SIGNAL(textEdited(QString)), this,
    SLOT(edited_Search_2(QString)) );
    tableOfBooks2 = new TableViewOfBase();
    connect(tableOfBooks2, SIGNAL(clicked(QModelIndex)),
    SLOT(slotClikTable_2(const QModelIndex)) );
    buttonWriteOff2 = new QPushButton("Списать книги");
    connect(buttonWriteOff2, SIGNAL(released()),
    SLOT(buttonWriteOff2_Released()) );
    tabLayout2->addWidget(comboBookPopularity2);
    tabLayout2->addWidget(lineSearch2);
    tabLayout2->addWidget(tableOfBooks2);
    tabLayout2->addWidget(buttonWriteOff2);
    tabLayout2->setAlignment(buttonWriteOff2,Qt::AlignCenter);
    tab2->setLayout(tabLayout2);
    activateTab_2(false);
    updateTableOfPopularity(comboBookPopularity2->currentIndex(),lineSearch1-
>text());
}
void MainWindow::createTab3()
{
    //tab3
    tab3 = new QWidget();
    tabLayout3 = new QVBoxLayout();
    lineSearch3 = new QLineEdit();
    lineSearch3->setPlaceholderText("поиск по таблице");
    connect(lineSearch3, SIGNAL(textEdited(QString)), this,
    SLOT(edited_Search_3(QString)) );
    tableOfBooks3 = new TableViewOfBase();
    tabLayout3->addWidget(lineSearch3);
    tabLayout3->addWidget(tableOfBooks3);
    tab3->setLayout(tabLayout3);
    updateTableOfBooks(lineSearch3->text());
}
void MainWindow::updatetableOfReaders(int numRequest, QString search)//
{
    tableOfReaders1->updateData(1,numRequest,search);
}
void MainWindow::updateTableOfPopularity(int numRequest, QString search)
{
    tableOfBooks2->updateData(2,numRequest,search);
}
void MainWindow::updateTableOfBooks(QString search)
{
    tableOfBooks3->updateData(3,0,search);
}
//--tab 1
void MainWindow::Reader_IndexChanged(int index)//combo
{
    qDebug() <<index;
    updatetableOfReaders(index,lineSearch1->text());
    activateTab_1(false);
}

```

```

void MainWindow::Action_IndexChanged(int index)//combo
{
}
//tab 2
void MainWindow::BookPopularity_IndexChanged(int index)//combo
{
    updateTableOfPopularity(index,lineSearch2->text());
    activateTab_2(false);
}
//--tab 1
void MainWindow::edited_Search_1(QString s)
{
    updatetableOfReaders(comboReaderGroup1->currentIndex(), s);
    activateTab_1(false);
}
//tab 2
void MainWindow::edited_Search_2(QString s)
{
    updateTableOfPopularity(comboBookPopularity2->currentIndex(), s);
    activateTab_2(false);
}
//tab 3
void MainWindow::edited_Search_3(QString s)
{
    updateTableOfBooks(s);
}
void MainWindow::activateTab_1(bool flag)//активировать элементы управления
{
    comboAction1->setEnabled(flag);
    buttonAgree1->setEnabled(flag);
    if(!flag)
        selectedReaderId="";
}
void MainWindow::activateTab_2(bool flag)//активировать элементы управления
{
    buttonWriteOff2->setEnabled(flag);
    if(!flag)
        selectedBookISBN="";
}
void MainWindow::slotClikTable_1(const QModelIndex &index)//выбран читатель
{
    int row = index.row();
    TableViewOfBase* thisTable = ((TableViewOfBase*)sender());//
        selectedReaderId = (thisTable->model()->data(thisTable->model()-
>index(row,0))).toString();
    //запустить отображение вариантов действий, и кнопки продолжить
    activateTab_1(true);
}
void MainWindow::slotClikTable_2(const QModelIndex &index)
{
    int row = index.row();
    TableViewOfBase* thisTable = ((TableViewOfBase*)sender());//
        selectedBookISBN = (thisTable->model()->data(thisTable->model()-
>index(row,0))).toString();
    //запустить отображение кнопки "Списать" если это возможно
    if(comboBookPopularity2->currentIndex()==1)
    {
        activateTab_2(true);
    }
}
bool MainWindow::isCanIssue()
{//узнать количество книг у читателя
    QSqlQuery *query = BD::getInstance()->make( QString("select count(isbn)
from books_of_reader"

```



```

        " WHERE (id_reader=\'%1\') and
(ISNULL(date_of_return))").arg(selectedReaderId) );
    if(query == NULL)
        return true;
    query->next();
    if(query->value(0).toInt() < 5)
        return true;
    else
        return false;
}
void MainWindow::buttononAgree1_Released()
{
    Form_Book_Issue *book_Issue;
    Form_Book_Return *book_Return;
    Form_WriteOff_Books *verification;
    switch (comboAction1->currentIndex()) {
    case 0:
        if(!isCanIssue())
        {
            QMessageBox::warning(this, "Ошибка выдачи", "Читатель не может
иметь более 5 книг на руках", QMessageBox::Ok);
            break;
        }
        book_Issue = new Form_Book_Issue(selectedReaderId,this);//isbn
        connect(book_Issue,
SIGNAL(finished(int)) ,this,SLOT(book_IssueDestroy(int) ));
        book_Issue->show();
        break;
    case 1:
        book_Return = new Form_Book_Return(selectedReaderId,this);//isbn
        connect(book_Return,
SIGNAL(finished(int)) ,this,SLOT(book_ReturnDestroy(int) ));
        book_Return->show();
        break;
    case 2:
        verification = new Form_WriteOff_Books(selectedBookISBN,this);//isbn
        connect(verification,
SIGNAL(finished(int)) ,this,SLOT(verificationDestroy(int) ));
        verification->show();
        break;
    case 3:
        break;
    default:
        break;
    }
}
void MainWindow::buttonWriteOff2_Released()
{
    Form_WriteOff_Books *verification = new
Form_WriteOff_Books(selectedBookISBN,this);//isbn
    connect(verification,
SIGNAL(finished(int)) ,this,SLOT(verificationDestroy(int) ));
    verification->show();
}
void MainWindow::formLoginDestroy(int result)
{
    if(result==0)
    {
        buid();//размещение на форме всех элементов
        //this->setVisible(true);
    }
    else
        this->close();
}

```

```

void MainWindow::book_IssueDestroy(int result)
{
    if(result==0)//была выдана книга
    {
        updatetableOfReaders(comboReaderGroup1->currentIndex(), lineSearch1-
>text());
        activateTab_1(false);
        updateTableOfPopularity(comboBookPopularity2->currentIndex(),
lineSearch2->text());//обновление тыблицы
        activateTab_2(false);
        updateTableOfBooks(lineSearch3->text() );
    }
}
void MainWindow::book_ReturnDestroy(int result)
{
    if(result==0)//была возвращена книга
    {
        updatetableOfReaders(comboReaderGroup1->currentIndex(), lineSearch1-
>text());
        activateTab_1(false);
        updateTableOfPopularity(comboBookPopularity2->currentIndex(),
lineSearch2->text());//обновление тыблицы
        activateTab_2(false);
        updateTableOfBooks(lineSearch3->text() );
    }
}
void MainWindow::verificationDestroy(int result)
{
    if(result==0)//было выполнено списывание книг
    {
        updateTableOfPopularity(comboBookPopularity2->currentIndex(),
lineSearch2->text());//обновление тыблицы
        activateTab_2(false);
        updateTableOfBooks(lineSearch3->text() );
    }
}

```