МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«Новосибирский национальный исследовательский государственный университет»

(Новосибирский государственный университет)

Структурное подразделение Новосибирского государственного университета –

Высший колледж информатики Университета

КАФЕДРА ИНФОРМАТИКИ

**Разработка приложения для базы данных**

**«Библиотека»**

Студентка гр.807са1 Окулова В.С.

“\_\_\_\_”\_\_\_\_\_\_\_\_\_2019 г.

Новосибирск

2019

Оглавление

[Постановка задачи 3](#_Toc10840545)

[Структура базы данных 4](#_Toc10840546)

[Руководство пользователя 6](#_Toc10840547)

[Вход. 6](#_Toc10840548)

[Я – Пользователь 7](#_Toc10840549)

[Я – Сотрудник 9](#_Toc10840550)

[Заключение 19](#_Toc10840551)

# Постановка задачи

*Описание предметной области:*

Разработать информационное приложение, с помощью которого можно автоматизировать функции работника библиотеки:

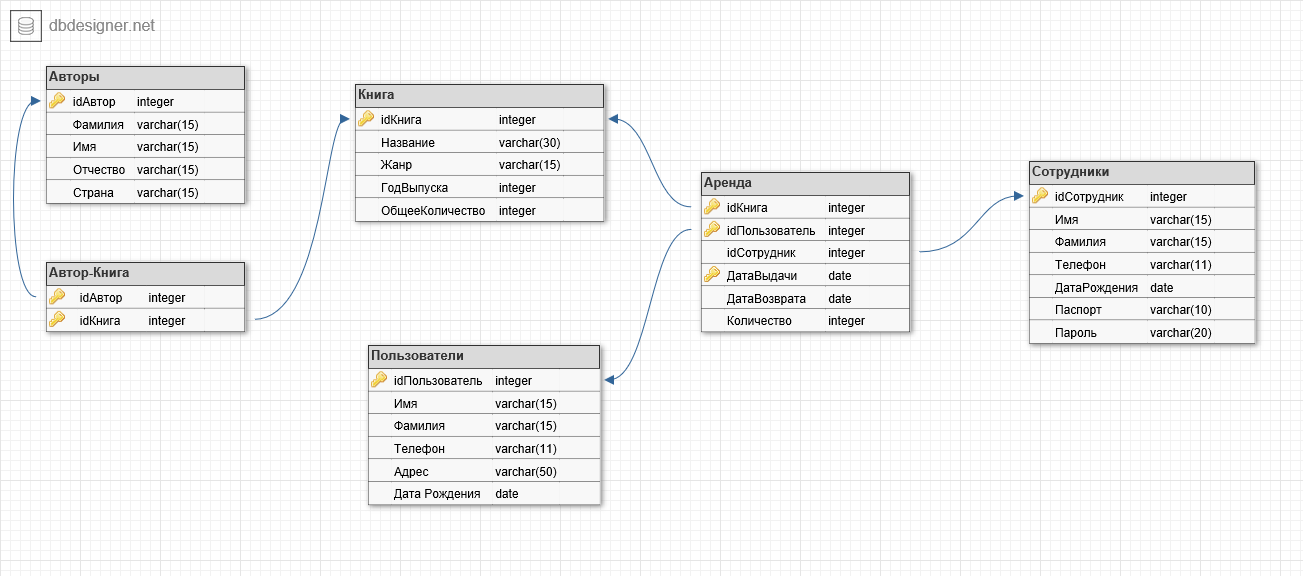
1. Ведение журнала пользователей библиотеки – в нём хранится вся необходимая информация о пользователе.
2. Ведение журнала о сотрудниках.
3. Ведение базы данных библиотеки, в которой находится вся информация о книгах, включая их количество.
4. Реализация поиска данных.
5. Реализация выдачи и приёма книг.

*Виды пользователей:*

1. Сотрудники библиотеки. Обладают полным функционалом: просмотром таблиц, добавлением книг, авторов, новых сотрудников и пользователей, изменением и удалением этой информации, а также могут выдавать и принимать книги.
2. Пользователи библиотеки. Обладают урезанным функциями: могут просматривать книги и авторов, а так же просматривать свой читательский билет по его ID(взятые книги, их количество, дата получения книги и дата, когда следует вернуть книгу)

# Структура базы данных

ER-модель



*Описание таблиц*

Сотрудники

|  |  |  |
| --- | --- | --- |
| *Имя столбца* | *Тип атрибута* | *Описание атрибута* |
| idСотрудника | Integer | Первичный ключ |
| Имя | Nvarchar(15) | Имя сотрудника |
| Фамилия | Nvarchar(15) | Фамилия сотрудника |
| Телефон | Varchar(11) | Номер телефона сотрудника |
| Дата Рождения | Date | Дата Рождения сотрудника |
| Паспорт | Varchar(10) | Паспорт сотрудника. Используется для входа в систему |
| Пароль | Nvarchar(20) | Пароль сотрудника. Используется для входа в систему |

Пользователи

|  |  |  |
| --- | --- | --- |
| *Имя столбца* | *Тип атрибута* | *Описание атрибута* |
| idПользователь | Integer | Первичный ключ |
| Имя | Nvarchar(15) | Имя пользователя |
| Фамилия | Nvarchar(15) | Фамилия пользователя |
| Телефон | Varchar(11) | Номер телефона пользователя |
| Адрес | Nvarchar(50) | Адрес пользователя |
| Дата Рождения | Date | Дата Рождения пользователя |

Авторы

|  |  |  |
| --- | --- | --- |
| *Имя столбца* | *Тип атрибута* | *Описание атрибута* |
| idАвтор | Integer | Первичный ключ |
| Фамилия | Nvarchar(15) | Фамилия автора |
| Имя | Nvarchar(15) | Имя автора |
| Отчество | Nvarchar(15) | Отчество автора |
| Страна | Nvarchar(15) | Страна автора |

Книги

|  |  |  |
| --- | --- | --- |
| *Имя столбца* | *Тип атрибута* | *Описание атрибута* |
| idКнига | Integer | Первичный ключ |
| Название | Nvarchar(30) | Название книги |
| Жанр | Nvarchar(15) | Жанр книги |
| Год Выпуска | Integer | Год выпуска книги |
| Общее Количество | Integer | Количество экземпляров книги |

Автор-Книга

|  |  |  |
| --- | --- | --- |
| *Имя столбца* | *Тип атрибута* | *Описание атрибута* |
| idКнига | Integer | Первичный ключ, Внешний ключ(ссылка на таблицу Книги, на атрибут idКнига) |
| idАвтор | Integer | Первичный ключ, Внешний ключ(ссылка на таблицу Авторы, на атрибут idАвтор) |

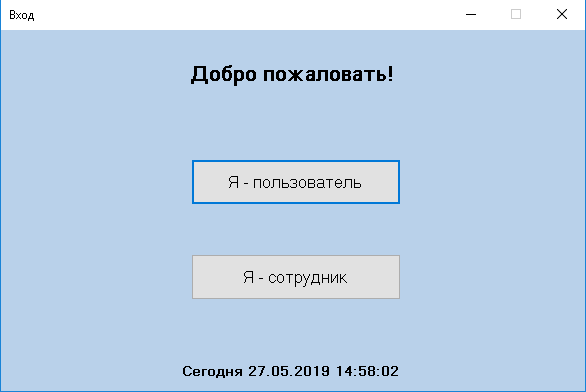
Аренда

|  |  |  |
| --- | --- | --- |
| *Имя столбца* | *Тип атрибута* | *Описание атрибута* |
| idКнига | Integer | Первичный ключ, Внешний ключ(ссылка на таблицу Книги, на атрибут idКнига) |
| idПользователь | Integer | Первичный ключ, Внешний ключ(ссылка на таблицу Пользователи, на атрибут idПользователь) |
| idСотрудник | Integer | Первичный ключ, Внешний ключ(ссылка на таблицу Сотрудники, на атрибут idСотрудник) |
| Дата Выдачи | Date | Дата выдачи книги |
| Дата Возврата | Date | Дата, когда следует вернуть книгу |
| Общее Количество | Integer | Количество взятых книг |

# Руководство пользователя

## Вход.

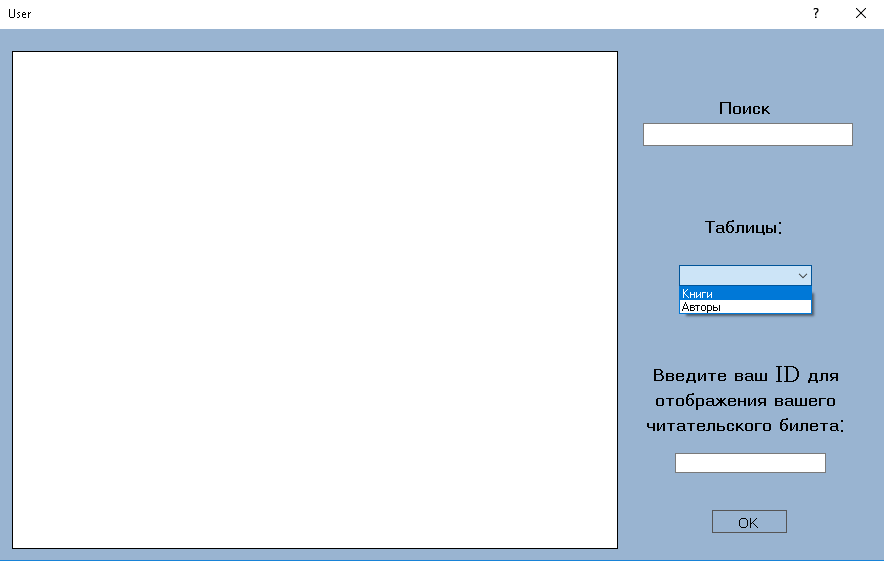
После запуска приложения появится следующее окно:



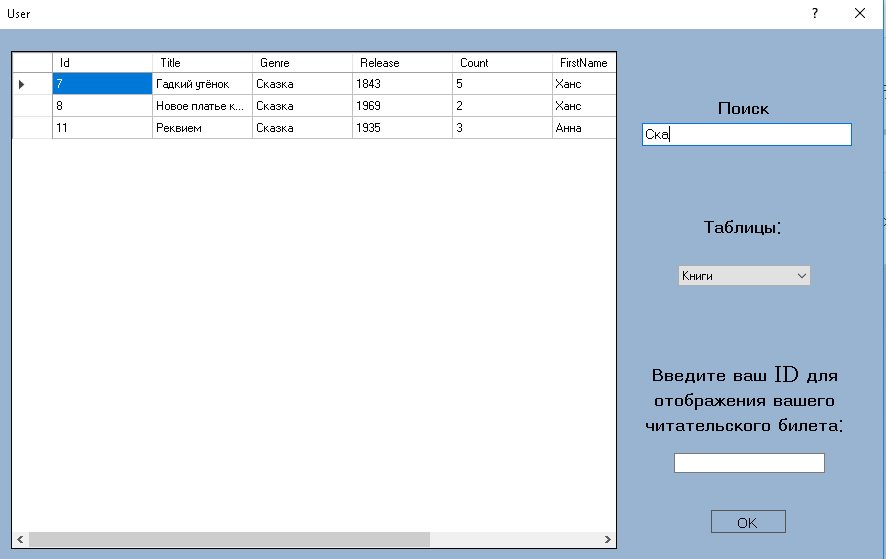
Для того, чтобы войти как пользователь, нужно нажать на кнопку “Я - пользователь”, а, чтобы войти как сотрудник, нужно нажать кнопку “Я - сотрудник”. На окне так же присутствуют часы, показывающие текущие дату и время компьютера.

## Я – Пользователь

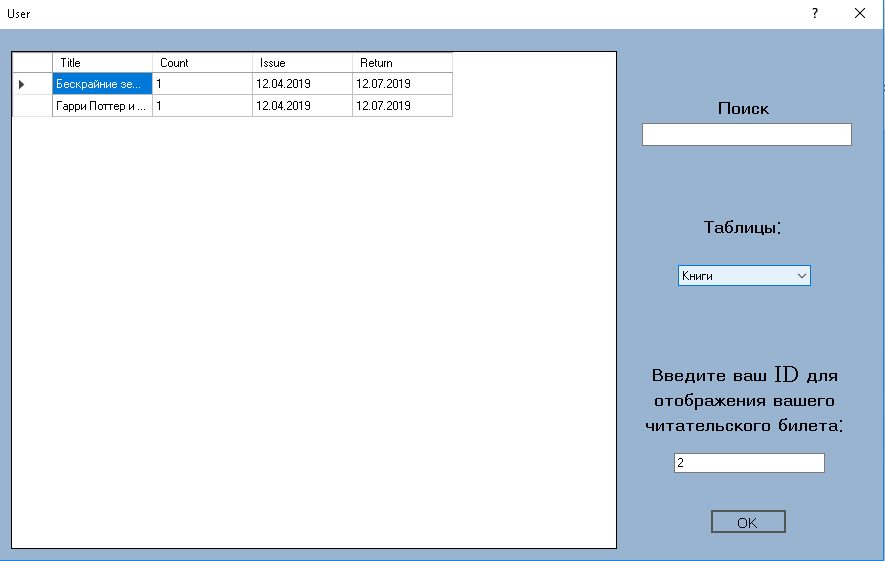
После нажатия кнопки “Я – пользователь” появляется новое окно, в котором находятся все возможности пользователя:



В данном окне можно просматривать данные из таблиц ‘Книги’ и ‘Авторы’. Для этого в меню справа из выпадающего меню нужно выбрать соответствующую таблицу, и она отобразится на панели слева.

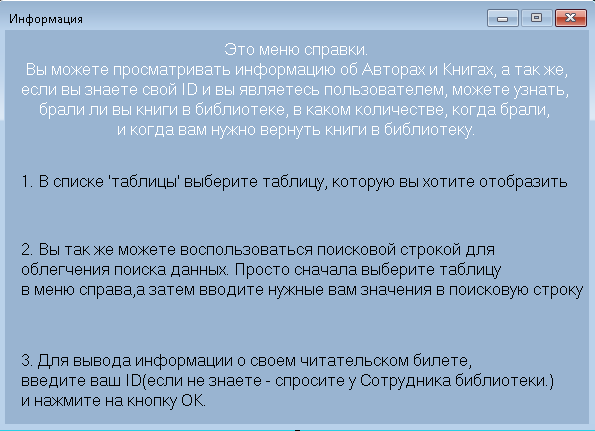


После этого можно пользоваться поисковой строкой, просто вводя нужные значения. Поиск осуществляется по принципу ‘Начинается с’.



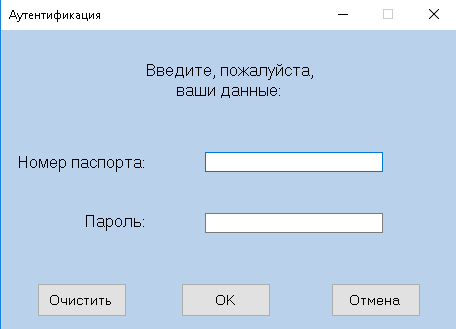
Для отображения своего читательского билета следует ввести целочисленное значение, больше 0 и нажать на кнопку ‘ОК’.

На окне, в шапке, в верхнем правом углу есть меню справки (знак ?), где так же написано краткое руководство как пользоваться данным окном:

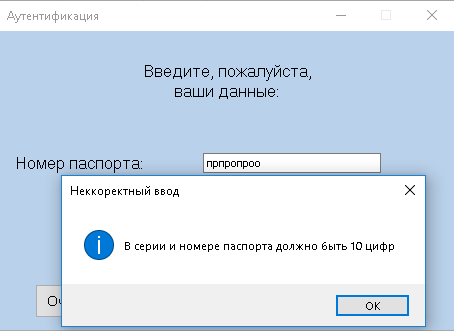


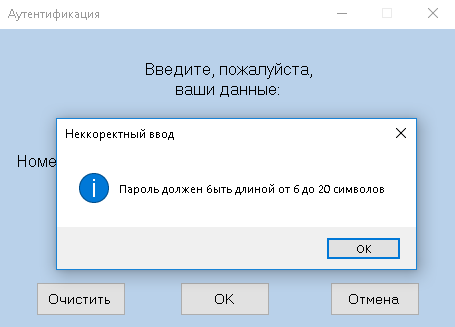
## Я – Сотрудник

После нажатия кнопки ”Я – Сотрудник” появится окно «Аутентификация», в котором нужно выполнить вход в систему, введя правильные паспорт и пароль:

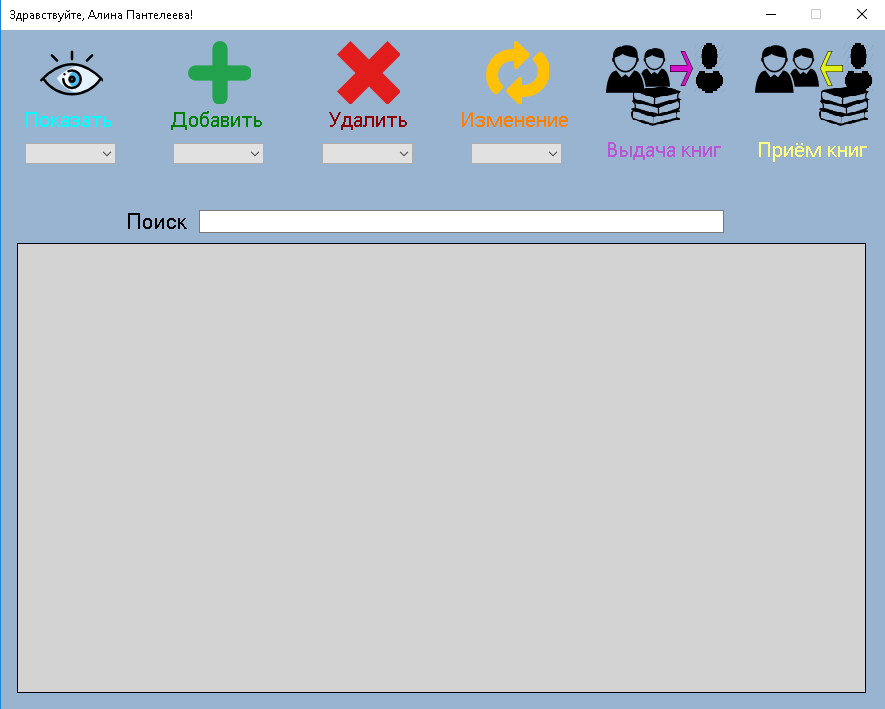


Номер и серия паспорта должны содержать только 10 цифр подряд, а пароль имеет ограничение лишь на длину – от 6 до 20 знаков. При нарушении какого-то из условий будет отображено соответствующее окно:





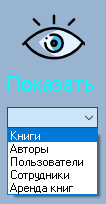
После ввода верных данных отображается главное окно:



Для облегчения поиска данных в отображаемых таблицах на окне присутствует поисковая строка, производящая поиск по принципу «начинается с»

1. Показать.

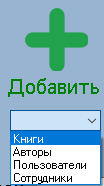
Для того, чтобы отобразить нужную таблицу, нужно выбрать её из меню ниже:



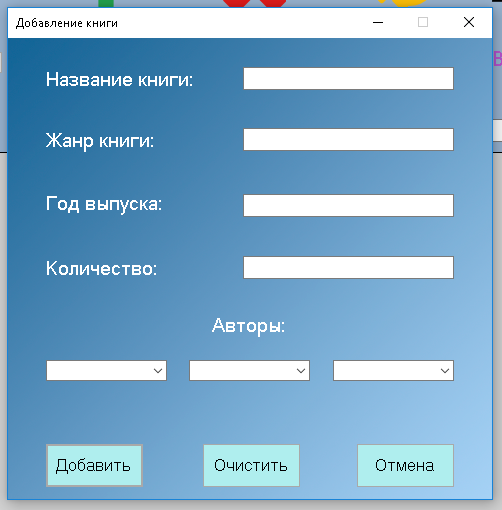
И после нажать по кнопке в виде глаза для отображения нужной таблицы.

1. Добавить

Для добавления новых записей в таблицы, в меню ниже нужно выбрать, в какую из таблиц следует добавить данные, а затем нажать на кнопку в виде знака ‘+’:



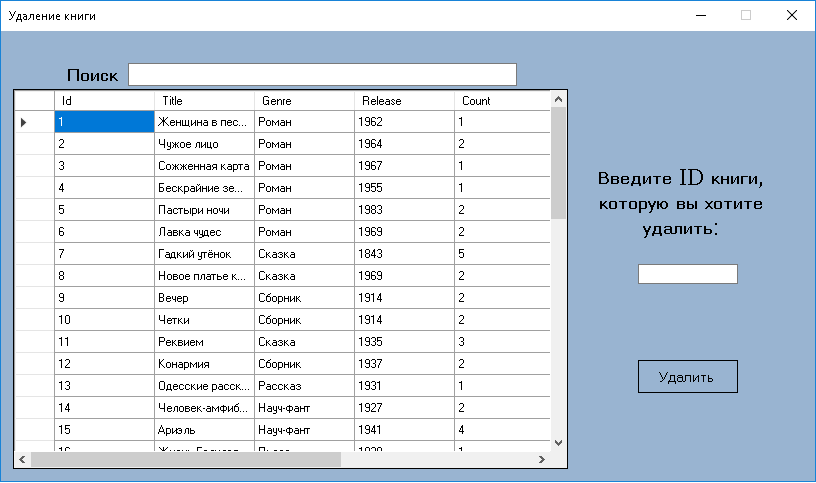
После этого будет показано соответствующее окно, в котором можно добавить данные. Так, к примеру, будет выглядеть окно, если добавлять данные в ‘Книги’:



Нужно заполнить все поля (год выпуска – необязательно) и корректно. При некорректном заполнении будут показаны соответствующие окна. Чтобы добавить новую запись, нужно нажать на кнопку ‘Добавить’. С помощью кнопки ‘Очистить’ можно очистить все поля на окне.

Для других таблиц всё выглядит аналогично, изменятся лишь необходимые данные для добавления.

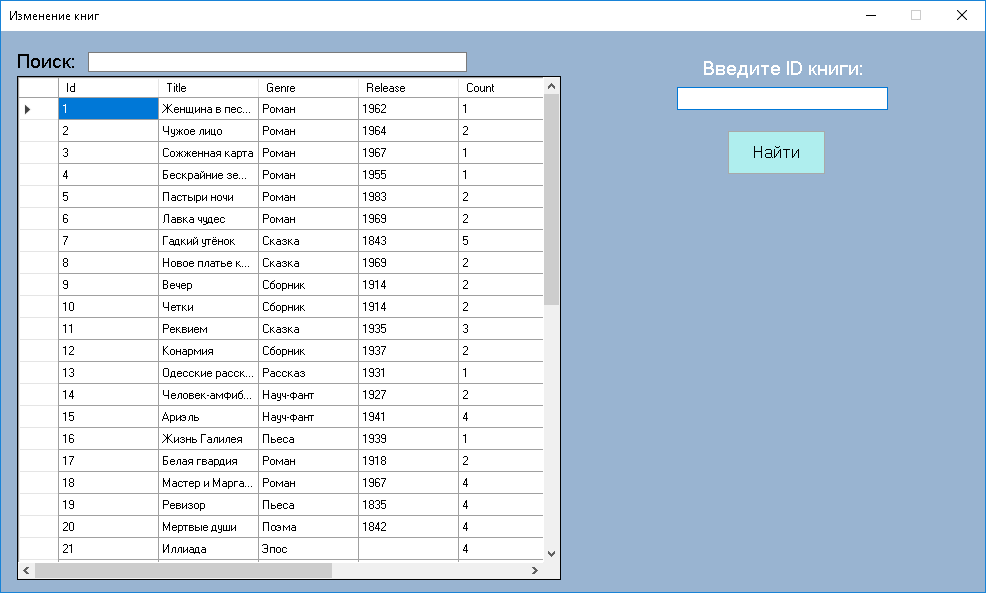
1. Удалить

Аналогично, нужно сначала выбрать таблицу в меню ниже, а затем нажать на кнопку ‘Удалить’ в виде крестика. Для таблицы ‘Книги’ окно выглядит следующим образом:

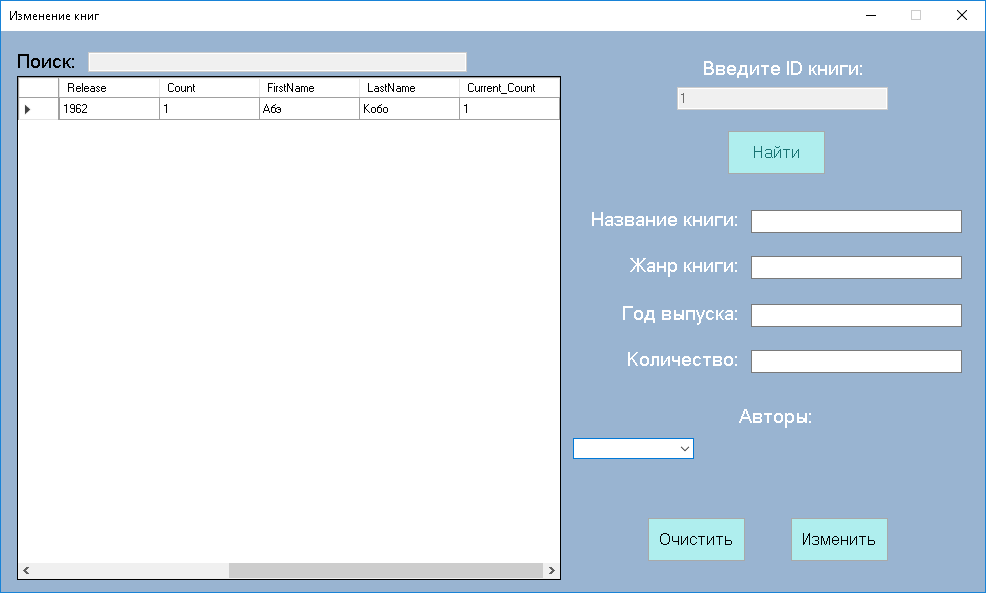
Для удобства поиска присутствует поисковая строка. Для того, чтобы удалить все сведения о книге (включая аренду книги), нужно в поле ввести её id (первый столбец в таблице) и нажать на кнопку ‘удалить’. Аналогично и для других таблиц.

1. Изменение

Аналогично, нужно сначала выбрать таблицу в меню ниже, а затем нажать на кнопку ‘Изменение’ в виде двух стрелочек. Для таблицы ‘Книги’ окно выглядит следующим образом:



Чтобы изменить данные о книге, нужно ввести в поле ввода её ID (первый столбец в таблице) и нажать на кнопку ‘Найти’. Для удобства поиска данных можно воспользоваться поисковой строкой.

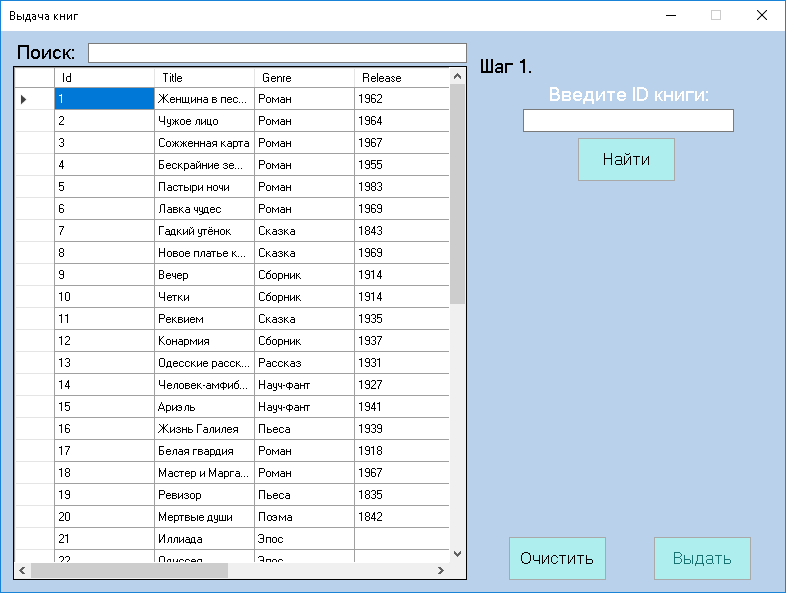


В окне присутствуют все данные, которые можно изменить. Если какое-то поле менять не нужно, следует оставить его пустым. Изменить можно до 3-х авторов, количество авторов, которых можно изменить равно изначальному количеству авторов. Так, если вы, когда добавляли книгу, указали только одного автора, то менять можно только его.

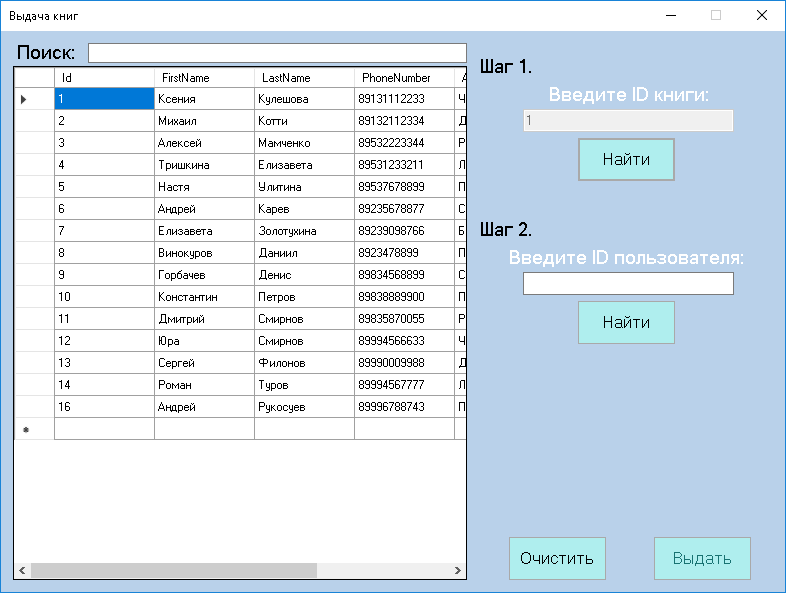
После заполнения следует нажать на кнопку ‘Изменить’. Кнопка ‘Очистить’ вернёт окно в изначальное состояние.

1. Выдача книг.

После нажатия на кнопку ‘Выдача книг’ появляется следующее окно:

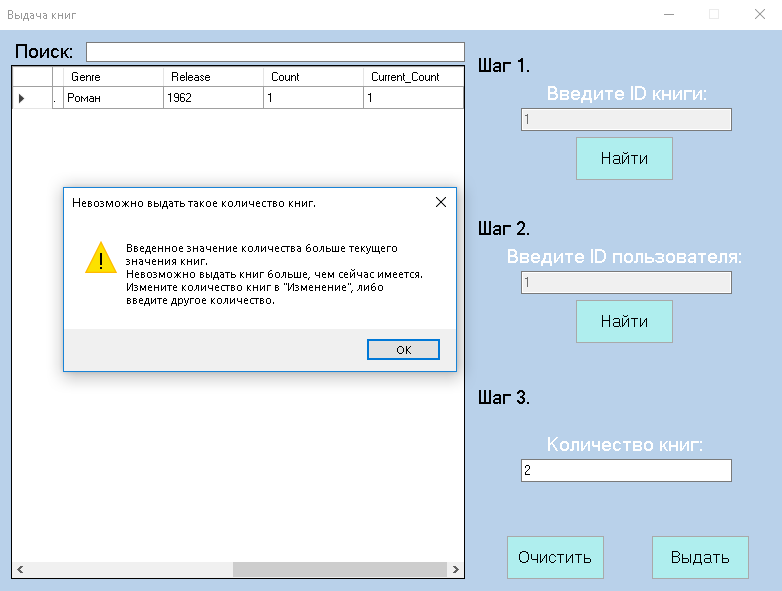


Процесс выдачи книги проходит в 3 последовательных шага, указанных на окне. 1 из них – ввод ID книги, которую нужно выдать. Для облегчения поиска по данным на окне присутствует поисковая строка.



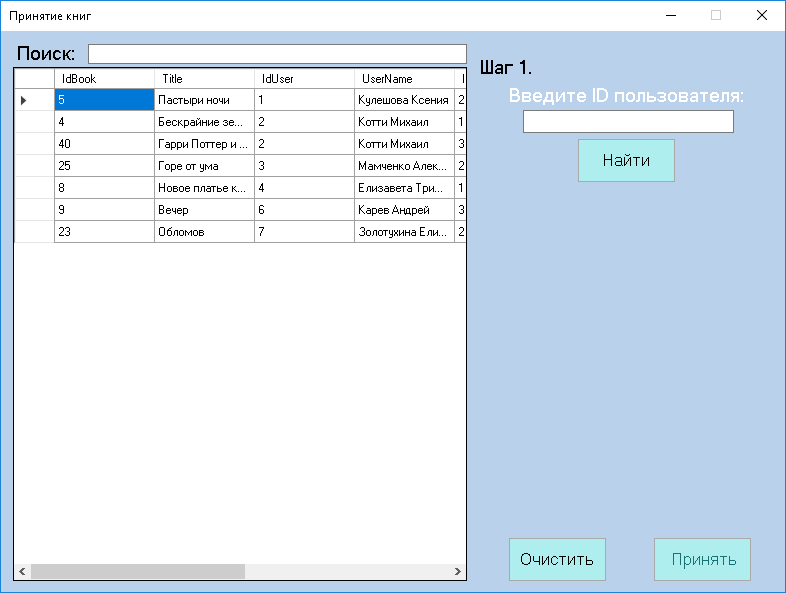
2 шаг – ввод ID пользователя, которому нужно выдать книгу.

И 3 шаг – ввод количества выдаваемых книг. При этом нужно ввести число, не большее чем значение в столбце Current\_Count – текущем количестве этой книги



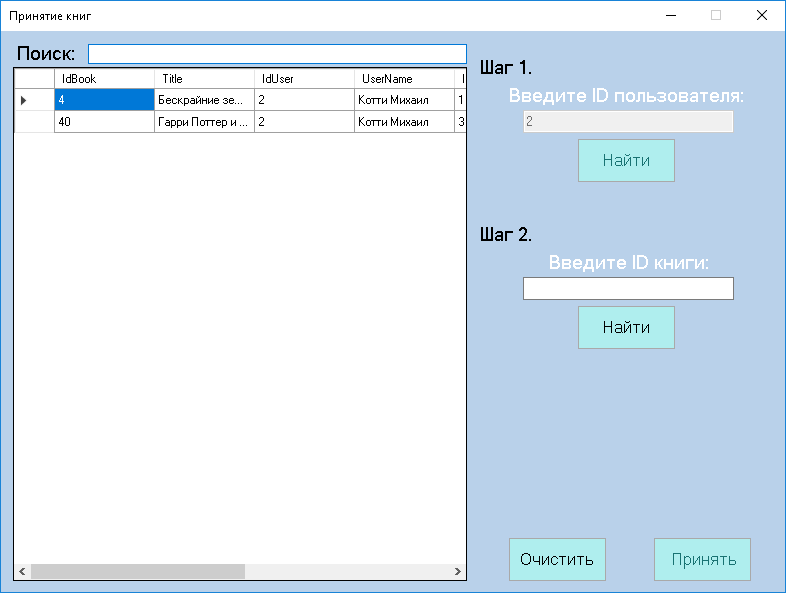
Чтобы выдать книгу, нажмите кнопку ‘Выдать’.

1. Приём книг.

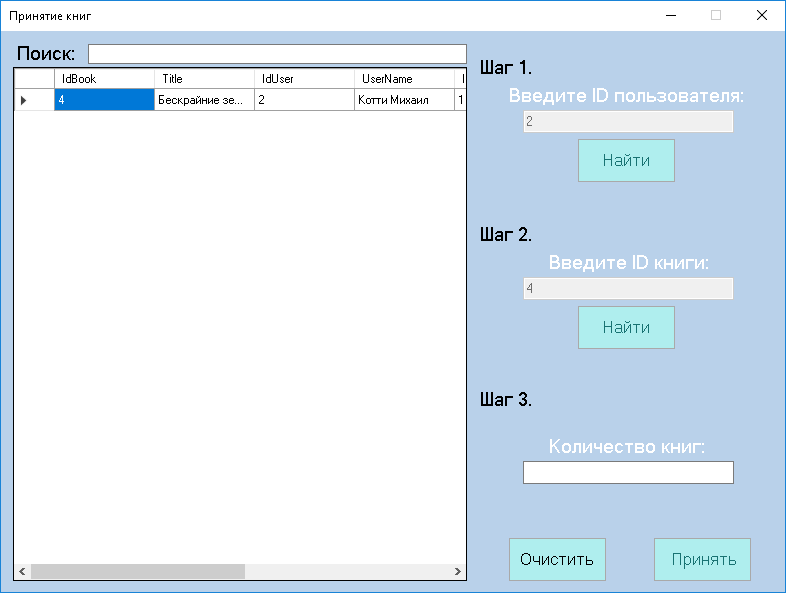


При нажатии на кнопку ‘Приём книг’ покажется окно, в котором отображены все выданные книги. Для облегчения поиска данных присутствует поисковая строка. Процесс возврата книги проходит в 3 шага.

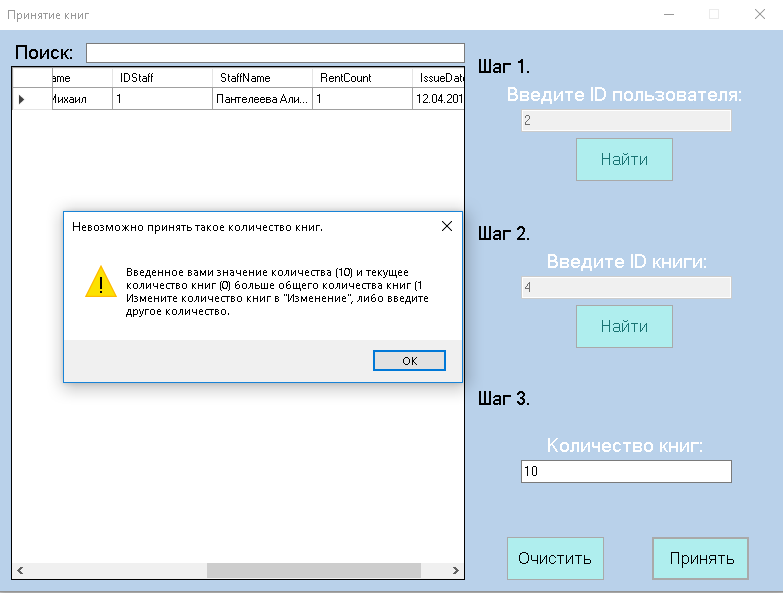
1 шаг – ввод ID пользователя (3 столбец)



2 шаг – ввод ID книги (1 столбец)



3 шаг – ввод количества возвращаемых книг. При этом нельзя вернуть такое количество книг, при котором текущее количество + количество возвращаемых книг станет больше общего количества книг:



# Заключение

В данной курсовой работе была выполнена реализация приложения для работы с библиотекой:

* Спроектирована база данных библиотеки;
* Реализована система входа в приложение для пользователей и сотрудников, при этом у каждого типа разные возможности: так, пользователи могут лишь просматривать данные двух таблиц, а также просматривать свой читательский билет, когда сотрудники наделены всеми возможности приложения;
* Реализована возможность работы с базой данных (добавление, удаление, изменение данных), а также удобная поисковая строка;
* Осуществлены главные функции работника библиотеки: приём и выдача книг.

Таким образом, все поставленные задачи были выполнены, однако приложение можно улучшить:

1. Улучшив и упростив дизайн- к примеру, главное меню сотрудников можно было сделать через компонент MenuStrip, адаптировать масштабы под экраны разного разрешения и др.;
2. Более грамотно реализовав добавление (возможность добавлять сколь угодно авторов

и изменение книг (возможность изменять сколь угодно авторов и добавлять новых);

1. Добавив различные статистические запросы (какие книги больше всего берут, кто берет больше книг, список должников и многие другие);
2. Добавление нового пользователя приложения: администратора. Так, сотрудники смогут лишь добавлять, удалять и изменять сведения только о пользователях библиотеки, а также выдавать и принимать книги, тогда как администраторы – все текущие возможности сотрудников в разработанном приложении.
3. Реализовать многопользовательность, при этом улучшив базу данных для сохранения её целостности.

# Листинг кода

## Папка entities

### Класс ‘Книга’

using System.Data.Linq.Mapping;

namespace Okulova\_CourseWork.entities

{

[Table(Name = "Книга")]

public class Books

{

[Column(Name = "idКнига", IsPrimaryKey = true, IsDbGenerated = true)]

public int Id { get; set; }

[Column(Name = "Название")]

public string Title { get; set; }

[Column(Name = "Жанр")]

public string Genre { get; set; }

[Column(Name = "Год Выпуска")]

public int? Release { get; set; }

[Column(Name = "Общее количество")]

public int Count { get; set; }

}

}

### Класс ‘Авторы’

using System.Data.Linq.Mapping;

namespace Okulova\_CourseWork.entities

{

[Table(Name = "Авторы")]

public class Authors

{

[Column(Name = "idАвтор", IsPrimaryKey = true, IsDbGenerated = true)]

public int Id { get; set; }

[Column(Name = "Фамилия")]

public string LastName { get; set; }

[Column(Name = "Имя")]

public string FirstName { get; set; }

[Column(Name = "Отчество")]

public string Patronymic { get; set; }

[Column(Name = "Страна")]

public string Country { get; set; }

}

}

### Класс ‘Пользователи’

using System;

using System.Data.Linq.Mapping;

namespace Okulova\_CourseWork.entities

{

[Table(Name = "Пользователи")]

class Users

{

[Column(Name = "idПользователь", IsPrimaryKey = true, IsDbGenerated = true)]

public int Id { get; set; }

[Column(Name = "Имя")]

public string FirstName { get; set; }

[Column(Name = "Фамилия")]

public string LastName { get; set; }

[Column(Name = "Телефон")]

public string PhoneNumber { get; set; }

[Column(Name = "Адрес")]

public string Adress { get; set; }

[Column(Name = "ДатаРождения")]

public DateTime Birthday { get; set; }

}

}

### Класс ‘Сотрудники’

using System;

using System.Data.Linq.Mapping;

namespace Okulova\_CourseWork.entities

{

[Table(Name = "Сотрудники")]

class Staff

{

[Column(Name = "idСотрудник", IsPrimaryKey = true, IsDbGenerated = true)]

public int Id { get; set; }

[Column(Name = "Имя")]

public string FirstName { get; set; }

[Column(Name = "Фамилия")]

public string LastName { get; set; }

[Column(Name = "Телефон")]

public string PhoneNumber { get; set; }

[Column(Name = "ДатаРождения")]

public DateTime Birthday { get; set; }

[Column(Name = "Паспорт")]

public string Passport { get; set; }

[Column(Name = "Пароль")]

public string Password { get; set; }

}

}

### Класс ‘Аренда’

using System;

using System.Data.Linq.Mapping;

namespace Okulova\_CourseWork.entities

{

[Table(Name = "Аренда")]

class Rent

{

[Column(Name = "idСотрудник")]

public int IdStaff { get; set; }

[Column(Name = "idПользователь", IsPrimaryKey = true)]

public int IdUsers { get; set; }

[Column(Name = "idКнига", IsPrimaryKey = true)]

public int IdBooks { get; set; } //

[Column(Name = "Дата Выдачи", IsPrimaryKey = true)]

public DateTime Issue { get; set; }

[Column(Name = "Дата Возврата", CanBeNull = true)]

public DateTime? Return { get; set; }

[Column(Name = "Количество", CanBeNull = false)]

public int? Count { get; set; }

}

}

### Класс ‘Автор-Книга’

using System.Data.Linq.Mapping;

namespace Okulova\_CourseWork.entities

{

[Table(Name = "Автор-книга")]

public class Author\_Book

{

[Column(Name = "idАвтор", IsPrimaryKey = true)]

public int IdAuthors { get; set; }

[Column(Name = "idКнига", IsPrimaryKey = true)]

public int IdBooks { get; set; }

}

}

## Папка dbHelper. Класс DB

using Okulova\_CourseWork.entities;

using System.Data.Linq;

using System.Linq;

using System;

namespace Okulova\_CourseWork.dbHelper

{

class DB

{

private static readonly string \_connectionString =

@"Data Source=.\SQLEXPRESS;Initial Catalog=OKULOVA\_KURSACH;Integrated Security=True";

private readonly DataContext Dc = new DataContext(\_connectionString);

public Table<Authors> GetAuthorsTable()

{

return Dc.GetTable<Authors>();

}

public Table<Users> GetUsersTable()

{

return Dc.GetTable<Users>();

}

public Table<Staff> GetStaffTable()

{

return Dc.GetTable<Staff>();

}

public Table<Books> GetBooksTable()

{

return Dc.GetTable<Books>();

}

public Table<Rent> GetRentTable()

{

return Dc.GetTable<Rent>();

}

public Table<Author\_Book> GetAuthor\_BooksTable()

{

return Dc.GetTable<Author\_Book>();

}

public IQueryable GetBooksWithAuthorsTable()

{

return

from c in GetBooksTable()

join Author\_Book in GetAuthor\_BooksTable()

on c.Id equals Author\_Book.IdBooks

join Author in GetAuthorsTable()

on Author\_Book.IdAuthors equals Author.Id

select new

{

IDКНИГИ = c.Id, c.Title, c.Genre,

c.Release, c.Count, Author.FirstName, Author.LastName,

Current\_Count = c.Count - ((from rent in GetRentTable()

where c.Id == rent.IdBooks

select rent.Count).Sum() ?? 0)

};

}

public IQueryable GetFullRentTable()

{

return

from rent in GetRentTable()

join book in GetBooksTable()

on rent.IdBooks equals book.Id

join staff in GetStaffTable()

on rent.IdStaff equals staff.Id

join user in GetUsersTable()

on rent.IdUsers equals user.Id

select new

{

IdBook = book.Id, book.Title,

IdUser = user.Id, UserName = user.LastName + " " + user.FirstName,

IDStaff = staff.Id, StaffName = staff.LastName + " " + staff.FirstName,

RentCount = rent.Count, IssueDate = rent.Issue,

};

}

public int Login(string Passport, string Password)

{

var query = from c in GetStaffTable()

where c.Password == Password && c.Password == Password && c.Password == c.Passport

select c.Id;

int idStaff = query.FirstOrDefault();

if (idStaff != 0)

return 0; // You can change your password....

else

{

query = from c in GetStaffTable()

where c.Password == Password && c.Passport == Passport

select c.Id;

idStaff = query.FirstOrDefault();

if (idStaff == 0)

return -1; //Wrong passport or password

else

return 1; // Found

}

}

public int GetIdStaff(string Passport, string Password)

{

var query = from c in GetStaffTable()

where c.Password == Password && c.Passport == Passport

select c.Id;

return query.FirstOrDefault();

}

public IQueryable Search(int Table, string s)

{

// 0 - books with authors and the current\_count table

// 1 - authors table

// 2 - users table

// 3 - staff's table

// 4 - Rent with staff, book and users table

switch(Table)

{

case 0:

return

from c in GetBooksTable()

join Author\_Book in GetAuthor\_BooksTable()

on c.Id equals Author\_Book.IdBooks

join Author in GetAuthorsTable()

on Author\_Book.IdAuthors equals Author.Id

where c.Title.StartsWith(s) || c.Genre.StartsWith(s) || Convert.ToString(c.Release).StartsWith(s)

|| Convert.ToString(c.Count).StartsWith(s) || Author.FirstName.StartsWith(s) || Author.LastName.StartsWith(s)

select new { c.Id, c.Title, c.Genre, c.Release, c.Count, Author.FirstName, Author.LastName,

Current\_Count = c.Count - ((from rent in GetRentTable()

where c.Id == rent.IdBooks

select rent.Count).Sum() ?? 0)

};

case 1:

return

from c in GetAuthorsTable()

where c.LastName.StartsWith(s) || c.FirstName.StartsWith(s)

|| c.Patronymic.StartsWith(s) || c.Country.StartsWith(s)

select c;

case 2:

return

from c in GetUsersTable()

where c.LastName.StartsWith(s) || c.FirstName.StartsWith(s)

|| c.PhoneNumber.StartsWith(s) || c.Adress.StartsWith(s) || Convert.ToString(c.Birthday).StartsWith(s)

select c;

case 3:

return

from c in GetStaffTable()

where c.LastName.StartsWith(s) || c.FirstName.StartsWith(s)

|| c.PhoneNumber.StartsWith(s)

select c;

default:

return

from rent in GetRentTable()

join book in GetBooksTable()

on rent.IdBooks equals book.Id

join staff in GetStaffTable()

on rent.IdStaff equals staff.Id

join user in GetUsersTable()

on rent.IdUsers equals user.Id

where book.Title.StartsWith(s) || user.LastName.StartsWith(s) || user.FirstName.StartsWith(s)

|| Convert.ToString(rent.Count).StartsWith(s) || Convert.ToString(rent.Issue).StartsWith(s)

select new

{

IdBook = book.Id, book.Title, IdUser = user.Id,

UserName = user.LastName + " " + user.FirstName,

IDStaff = staff.Id, StaffName = staff.LastName + " " + staff.FirstName,

RentCount = rent.Count, IssueDate = rent.Issue,

};

}

}

public void Addbook(string Title, string Genre, int? Release, int Count)

{

Books book1 = new Books

{

Title = Title,

Genre = Genre,

Release = Release,

Count = Count,

};

GetBooksTable().InsertOnSubmit(book1);

Dc.SubmitChanges();

}

public void AddAuthor\_Book(int idBook, int idAuthor)

{

Author\_Book authbook = new Author\_Book

{

IdAuthors = idAuthor,

IdBooks = idBook,

};

GetAuthor\_BooksTable().InsertOnSubmit(authbook);

Dc.SubmitChanges();

}

public void AddAuthor(string LastName, string FirstName, string Patronymic, string Country)

{

Authors author = new Authors

{

LastName = LastName,

FirstName = FirstName,

Patronymic = Patronymic,

Country = Country

};

GetAuthorsTable().InsertOnSubmit(author);

Dc.SubmitChanges();

}

public void AddUser(string FirstName, string LastName, string PhoneNumber, string Adress, DateTime Birthday)

{

Users user = new Users

{

FirstName = FirstName,

LastName = LastName,

PhoneNumber = PhoneNumber,

Adress = Adress,

Birthday = Birthday

};

GetUsersTable().InsertOnSubmit(user);

Dc.SubmitChanges();

}

public void AddStaff(string FirstName, string LastName, string PhoneNumber, DateTime Birthday, string Passport)

{

Staff staff = new Staff

{

FirstName = FirstName,

LastName = LastName,

PhoneNumber = PhoneNumber,

Birthday = Birthday,

Passport = Passport,

Password = Passport

};

GetStaffTable().InsertOnSubmit(staff);

Dc.SubmitChanges();

}

public void AddRent(int IdStaff, int IdUser, int IdBook, DateTime Issue, DateTime Return, int Count)

{

Rent rent = new Rent

{

Count = Count,

IdBooks = IdBook,

IdStaff = IdStaff,

IdUsers = IdUser,

Issue = Issue,

Return = Return,

};

GetRentTable().InsertOnSubmit(rent);

Dc.SubmitChanges();

}

public void DeleteBook(int idBook)

{

// deleting from Author\_Book table

var query = from c in GetAuthor\_BooksTable()

where c.IdBooks == idBook

select c;

foreach (var detail in query)

{

GetAuthor\_BooksTable().DeleteOnSubmit(detail);

}

Dc.SubmitChanges();

// deleting from Rent table

var query1 = from q in GetRentTable()

where q.IdBooks == idBook

select q;

foreach(var detail in query1)

{

GetRentTable().DeleteOnSubmit(detail);

}

Dc.SubmitChanges();

// deleting from Books table

var query2 = from q in GetBooksTable()

where q.Id == idBook

select q;

GetBooksTable().DeleteOnSubmit(query2.FirstOrDefault());

Dc.SubmitChanges();

}

public void DeleteAuthor(int IdAuthor)

{

var query = from c in GetAuthor\_BooksTable()

where c.IdAuthors == IdAuthor

select c;

foreach (var detail in query)

{

GetAuthor\_BooksTable().DeleteOnSubmit(detail);

}

Dc.SubmitChanges();

////////////////////////////////////

var query1 = from c in GetAuthorsTable()

where c.Id == IdAuthor

select c;

GetAuthorsTable().DeleteOnSubmit(query1.FirstOrDefault());

Dc.SubmitChanges();

}

public void DeleteUser(int IdUser)

{

// deleting from Rent table

var query1 = from q in GetRentTable()

where q.IdUsers == IdUser

select q;

foreach (var detail in query1)

{

GetRentTable().DeleteOnSubmit(detail);

}

Dc.SubmitChanges();

// deleting from Users table

var query2 = from q in GetUsersTable()

where q.Id == IdUser

select q;

GetUsersTable().DeleteOnSubmit(query2.FirstOrDefault());

Dc.SubmitChanges();

}

public void DeleteStaff(int IdStaff)

{

// deleting from Rent table

var query1 = from q in GetRentTable()

where q.IdStaff == IdStaff

select q;

foreach (var detail in query1)

{

GetRentTable().DeleteOnSubmit(detail);

}

Dc.SubmitChanges();

// deleting from Staff table

var query2 = from q in GetStaffTable()

where q.Id == IdStaff

select q;

GetStaffTable().DeleteOnSubmit(query2.FirstOrDefault());

Dc.SubmitChanges();

}

public void DeleteRent(int IdUser, int idBook, int count)

{

// Тут есть два случая:

// 1. Мы возвращаем не все книги, в таком случае нужно изменить count

// 2. Мы возвращаем все книги, в таком случае нужно удалить всю запись.

// deleting from Rent table

var query = from q in GetRentTable()

where q.IdUsers == IdUser && q.IdBooks == idBook

select q.Count;

// 2

if (query.FirstOrDefault() == count)

{

var query1 = from q in GetRentTable()

where q.IdUsers == IdUser && q.IdBooks == idBook

select q;

foreach (var detail in query1)

{

GetRentTable().DeleteOnSubmit(detail);

}

Dc.SubmitChanges();

}

else

{

var query1 = from q in GetRentTable()

where q.IdUsers == IdUser && q.IdBooks == idBook

select q;

if (query.FirstOrDefault() < count)

foreach(Rent rent in query1)

{

rent.Count = rent.Count - count;

}

}

}

public void ChangeAuthorForBook(int IdBook, int IdAuthor)

{

var query = from c in GetAuthor\_BooksTable()

where c.IdBooks == IdBook

select c;

foreach (Author\_Book a\_b in query)

{

a\_b.IdBooks = IdBook;

a\_b.IdAuthors = IdAuthor;

}

Dc.SubmitChanges();

}

public void Changebook(int ID, string Title, string Genre, int Release, int Count)

{

var query = from c in GetBooksTable()

where c.Id == ID

select c;

foreach (Books book in query)

{

if (Title != "") book.Title = Title;

if (Genre != "") book.Genre = Genre;

if (Release != 0) book.Release = Release;

if (Count != 0) book.Count = Count;

book.Id = ID;

}

Dc.SubmitChanges();

}

public void ChangeAuthor(int ID, string LastName, string FirstName, string Patronymic, string Country)

{

var query = from c in GetAuthorsTable()

where c.Id == ID

select c;

foreach (Authors author in query)

{

if (LastName != "") author.LastName = LastName;

if (FirstName != "") author.FirstName = FirstName;

if (Patronymic != "") author.Patronymic = Patronymic;

if (Country != "") author.Country = Country;

author.Id = ID;

}

Dc.SubmitChanges();

}

public void ChangeUser(int ID, string FirstName, string LastName, string PhoneNumber, string Adress, DateTime Birthday)

{

var query = from c in GetUsersTable()

where c.Id == ID

select c;

foreach (Users user in query)

{

if (LastName != "") user.LastName = LastName;

if (FirstName != "") user.FirstName = FirstName;

if (PhoneNumber != "") user.PhoneNumber = PhoneNumber;

if (Adress != "") user.Adress = Adress;

user.Birthday = Birthday;

user.Id = ID;

}

Dc.SubmitChanges();

}

public void ChangeStaff(int ID, string FirstName, string LastName, string PhoneNumber, DateTime Birthday, string Passport)

{

var query = from c in GetStaffTable()

where c.Id == ID

select c;

foreach (Staff staff in query)

{

if (LastName != "") staff.LastName = LastName;

if (FirstName != "") staff.FirstName = FirstName;

if (PhoneNumber != "") staff.PhoneNumber = PhoneNumber;

if (Passport != "") staff.Passport = Passport;

staff.Birthday = Birthday;

staff.Id = ID;

}

Dc.SubmitChanges();

}

public void ChangePassword(int b, string s)

{

var query = from c in GetStaffTable()

where c.Id == b

select c;

foreach (Staff staff in query)

{

staff.Birthday = query.FirstOrDefault().Birthday;

staff.FirstName = query.FirstOrDefault().FirstName;

staff.LastName = query.FirstOrDefault().LastName;

staff.Passport = query.FirstOrDefault().Passport;

staff.PhoneNumber = query.FirstOrDefault().PhoneNumber;

staff.Id = b;

}

Dc.SubmitChanges();

}

}

}