Министерство науки и высшего образования РФ

Федеральное государственное автономное образовательное

учреждение высшего образования

«Казанский (Приволжский) федеральный университет»

Институт Вычислительной математики и информационных технологий

Кафедра системного анализа и информационных технологий

Отчет по дисциплине «Технологии баз данных»

Студент 3 курса

Группа 09-941 Сарыймова Л.Н.

Казань-2021

ОГЛАВЛЕНИЕ

[Введение 3](#_Toc91461577)

[Описание реляционной модели данных. 4](#_Toc91461578)

[Запросы к базе данных 6](#_Toc91461579)

# **Введение**

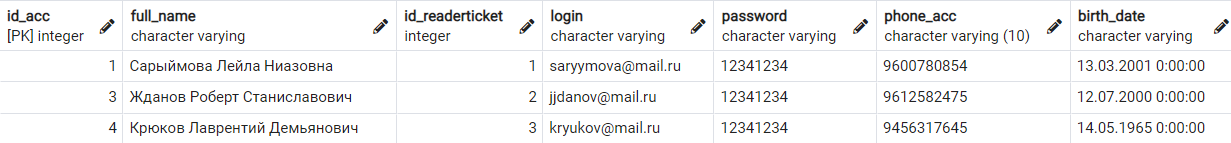
Семестровым заданием было создание клиентского приложения с базами данных. Базы данных – это информационная модель, содержащая в себе структуру данных о группе объектов в упорядоченном виде, которые обладают одинаковыми свойствами. Для своей работы я выбрала PostgresSQL. Это объектно-реляционная система управления базами данных. Она является альтернативой для многих коммерческих СУБД и прост в использовании, поэтому она и была выбрана мной для использования. Также приложение создавалось с помощью среды разработки оконных приложений Windows Forms на языке С#.

Моим проектом стало приложение для сети библиотек. С помощью него клиент может дистанционно посмотреть наличие нужной ему книги в выбранной библиотеке, заказать её, а получив её в библиотеке, может отслеживать дату ее сдачи. Такое приложение поможет автоматизировать работу библиотеки и снизить число людей в библиотеке, что особенно актуально в связи карантинными мерами. В нем будут содержаться данные, связанные с читателями, их читательскими билетами, книгами, библиотеками и издательствами.

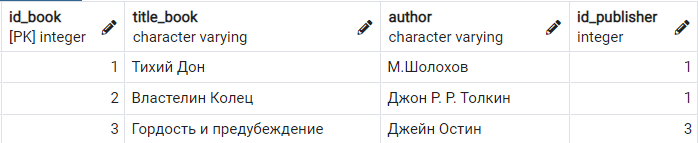
# **Описание реляционной модели данных.**

Данные в СУБД хранятся с помощью таблиц, в которой указаны характеристики столбцов. Я использовала следующие таблицы:

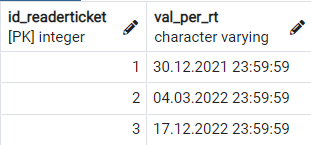
1) Таблица "accounts" содержит информацию о клиентах.



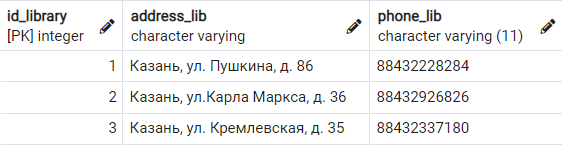
2) Таблица «books» содержит информацию о книгах и издательствах, в которых они были сделаны.



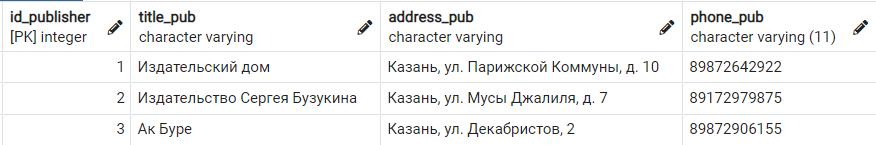
3) Таблица «reader\_tickets» содержит номера читательских билетов и их срок действия.



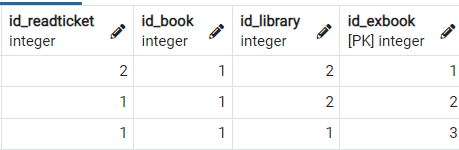
4) Таблица «libraries» содержит информацию о читательских билетах



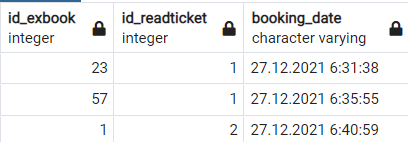
5) Таблица «publishers» содержит информацию об издательствах



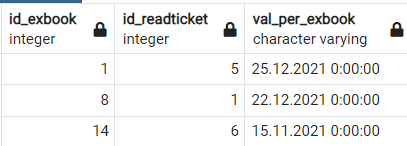
6) Таблица «exbooks» содержит информацию об экземплярах книг, о том, в какой библиотеке она находится и принадлежит ли какому-то читательскому билету.



7) Таблица «not\_taked\_books» содержит информацию об экземпляре книги, которую забронировали, читательском билете и дате, до которой книгу нужно забрать.



8) Таблица «taked\_books» содержит информацию об экземпляре книги, которую забрали, читательском билете и дате, когда ее нужно вернуть.



# **Запросы к базе данных**

**1. Проекция**

Вывести ФИО всех аккаунтов:

SELECT public."Accounts"."full\_name" FROM public."Accounts"

Вывести названия всех книг:

SELECT public."Books"."title\_book" FROM public."Books"

Вывести адреса всех библиотек:

SELECT public."Libraries"."address\_lib" FROM public." Libraries "

**2. Селекция**

Вывести названия книг начинающиеся с “Т”:

SELECT public."Books"."title\_book" FROM public."Books" WHERE "title\_book" LIKE 'Т%'

Вывести ФИО людей, чьи логины от Яндекс:

SELECT public."Accounts"."full\_name" FROM public."Accounts" WHERE "login" LIKE '%yandex%'

Вывести количество аккаунтов, которые младше 18:

SELECT count(\*) FROM public."Accounts" WHERE "birth\_date" > '2003-11-26';

**3. Объединение**

Вывести названия книг, которые написал Оруэлл и все книги, которые написал Шолохов:

SELECT public."Books"."title\_book" FROM public."Books" WHERE "author" LIKE '%Оруэлл%' UNION SELECT public."Books"."title\_book" FROM public."Books" WHERE "author" LIKE '%Шолохов%';

Вывести все аккаунты, у которых фамилия начинается на С и номер читательского билета у которых больше 5:

SELECT public."Accounts"."full\_name" FROM public." Accounts " WHERE "full\_name" LIKE '%C%' UNION SELECT public." Accounts "."id\_readerticket" FROM public." Accounts " WHERE "id\_readertickets" > 5;

Вывести экземпляры книг, которые не забронированы никем и название которых «Тихий Дон»:

SELECT public."Exbooks"."id\_exbook" FROM public."Exbooks" WHERE "id\_readerticket" = [null] UNION SELECT public."Exbooks"."title\_book" FROM public."Exbooks" WHERE "id\_book" = 1;

**4. Пересечение**

Вывести запросы, у которых логин от mail и у которых фамилия начинается на С:

SELECT public."Accounts"."full\_name" FROM public."Accounts" WHERE "login" LIKE '%mail%' AND "full\_name" LIKE 'С%'

Вывести книги, которые от Издательства «Издательский дом» и которые написаны Роулинг:

SELECT \* FROM public."Books" WHERE “id\_publisher” = 1 AND "author" LIKE '%Роулинг%';

**5. Разность**

Вывести экземпляры книг, которые из библиотеки 1 и автор которых не Адамс

SELECT \* FROM public."Exbooks" INNER JOIN public."Books" ON "Books"."id\_book" = "Exbooks"."id\_book" WHERE "id\_library" = 1 AND NOT "title\_book" LIKE '%Адамс%'

Вывести забронированные книги, идентификатор которых больше 50 и которые не забронированы человеком с читательским билетом 5:

SELECT \* FROM public."not\_taked\_books" WHERE "id\_exbook" > 50 AND NOT "id\_readertiket" = 5

**6. Соединение**

Вывести взятые книги, их обладателей и дату, когда их нужно вернуть:

SELECT public."Books"."title\_book", public."Accounts"."full\_name", "Taked\_books"."val\_per\_exbook" FROM "Taked\_books" INNER JOIN public."Exbooks" ON public."Exbooks"."id\_exbook" = "Taked\_books"."id\_exbook" JOIN public."Books" ON public."Books"."id\_book" = public."Exbooks"."id\_book" JOIN public."Accounts" ON "Taked\_books"."id\_readticket" = public."Accounts"."id\_readerticket"

**7. Деление**

Вывести аккаунты, у которых фамилия Иванов и имя Иван:

SELECT \* FROM "accounts" AS A1 WHERE NOT EXISTS (SELECT \* FROM "accounts" WHERE NOT EXISTS (SELECT \* FROM "accounts" AS A2 WHERE A2."full\_name" LIKE '%Иванов%' AND A2."full\_name" LIKE '%Иван%' AND A2."id\_film" = A1."id\_film"))

Вывести книги, автор которых Адамс и Пулман:

SELECT \* FROM "books" AS B1 WHERE NOT EXISTS (SELECT \* FROM "books" WHERE NOT EXISTS (SELECT \* FROM "books" AS B2 WHERE B2."title\_book" LIKE '%Адамс%' AND В2."Пулман" LIKE '%yandex%' AND A2."id\_film" = A1."id\_film"))

# **Серверные процедуры и функции**

**1. Функция для подсчета количества книг в конкретной библиотеке:**

CREATE FUNCTION BooksNumber (integer) RETURNS integer AS $$

DECLARE num INTEGER;

BEGIN

SELECT COUNT ("id\_exbook") FROM "exbooks" WHERE "id\_library"=$1 INTO num;

RETURN num;

END;

$$ LANGUAGE plpgsql;

**Пример вызова функции:**

select BooksNumber(1);

**2. Функция для подсчета количества книг, которые не забронировали, но не взяли до определенной даты:**

CREATE FUNCTION Not\_taked\_books\_number (date) RETURNS integer AS $$

DECLARE num INTEGER;

BEGIN

SELECT COUNT ("id\_exbook") FROM "not\_taked\_books" WHERE "booking\_date"<$1 INTO num;

RETURN num;

END;

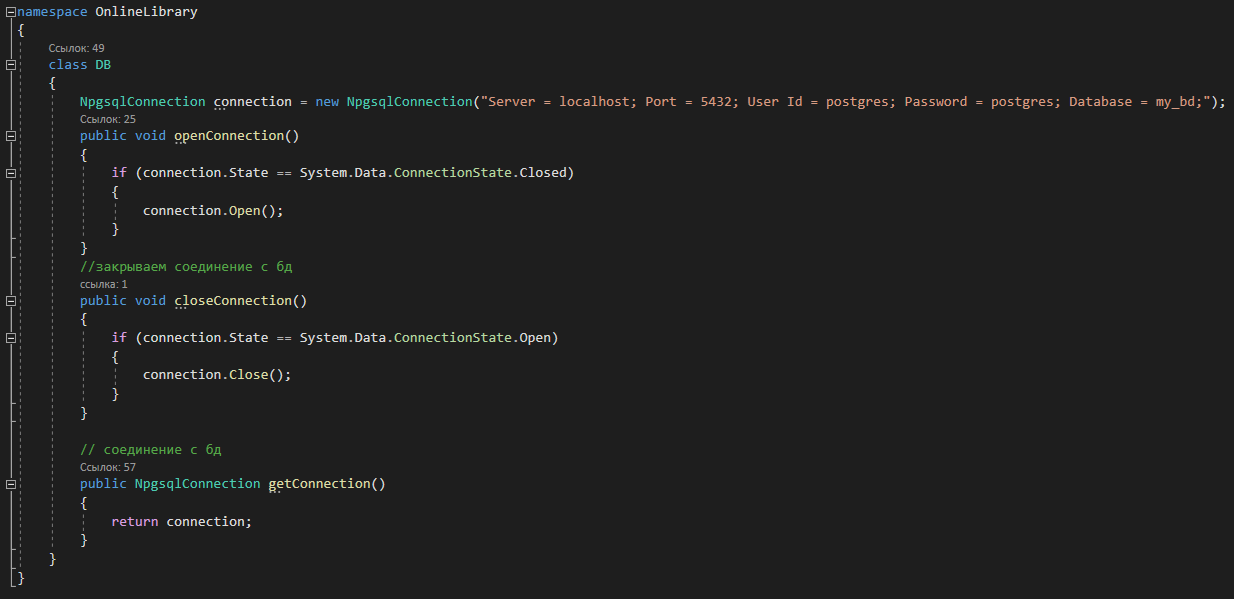
$$ LANGUAGE plpgsql;

**Пример вызова функции:**

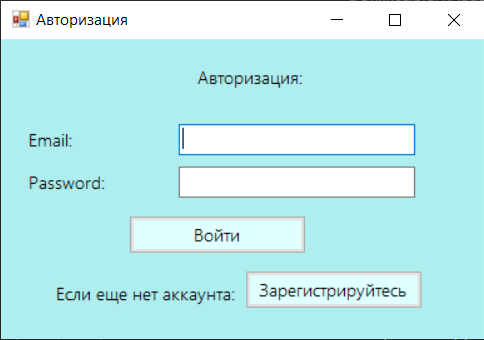
select Not\_taked\_books\_number (to\_date('2020-10-19', 'YYYY/MM/DD'));

# **Клиентское приложение**

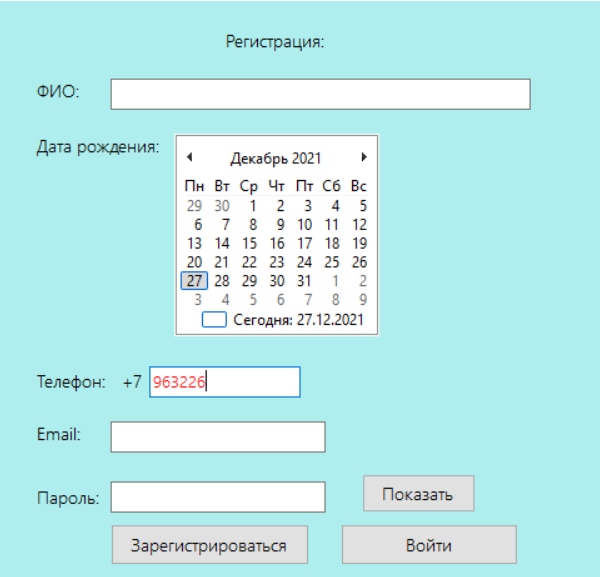
Для связи с приложением был создан отдельный класс BD.



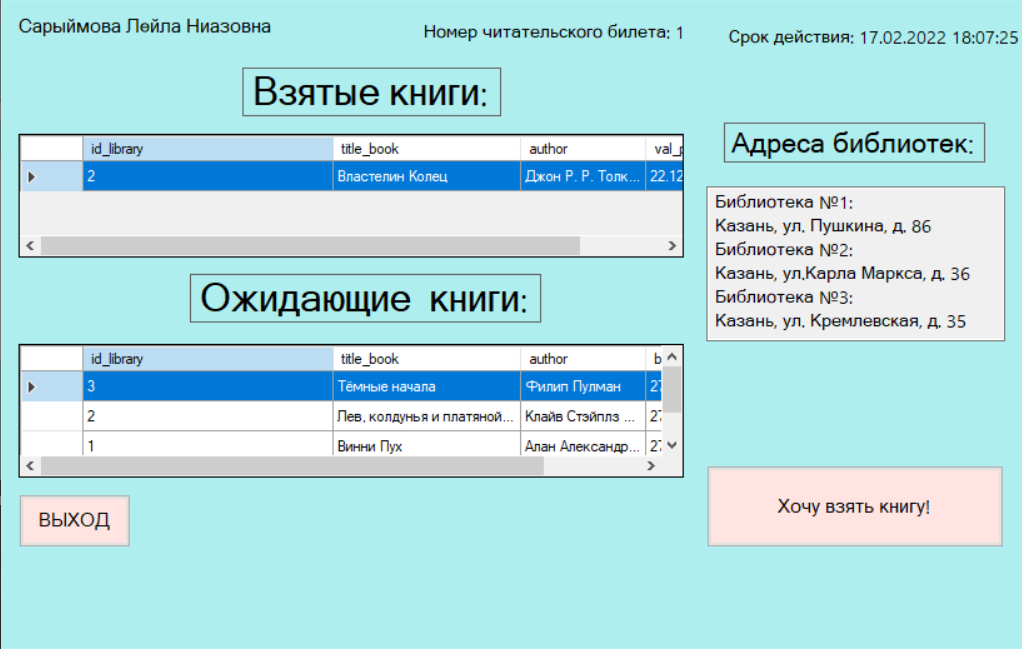
Первое, что видит клиент, заходя в приложение, предложение войти в свой аккаунт:



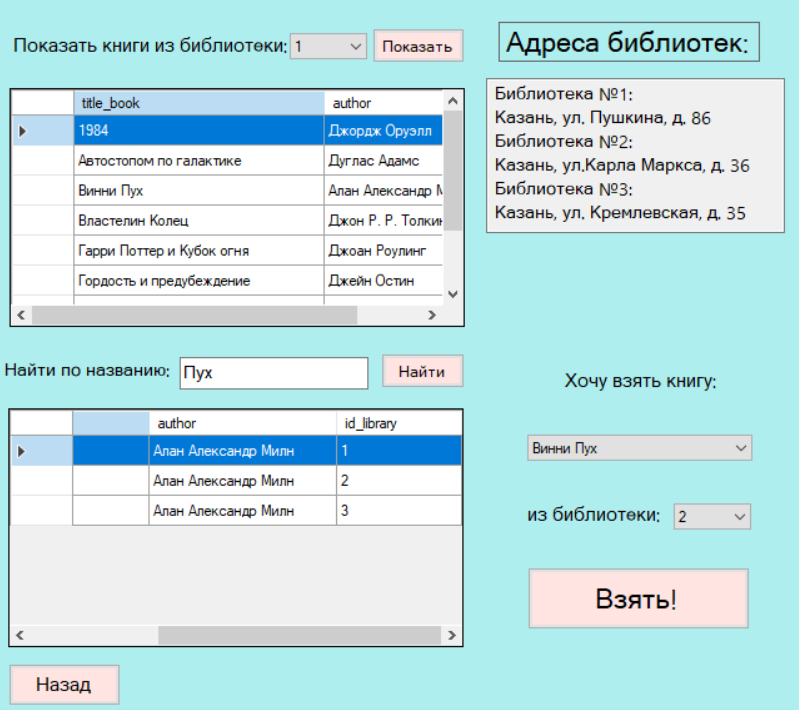
Во время регистрации человек вводит свои данные, причем приложение уведомляет, если аккаунт с выбранным E-mail’ом уже существует. Если количество цифр в номере не 10, то они подсвечиваются красным. При всех этих случаях зарегистрироваться невозможно. При желании можно посмотреть свой пароль, нажав на кнопку. Если же все гладко, сначала в таблице с читательскими билетами создается билет со сроком 2 месяца (на такое время дается бесплатное пользование). А уже затем в таблицу аккаунтов добавляются данные о новом пользователе.



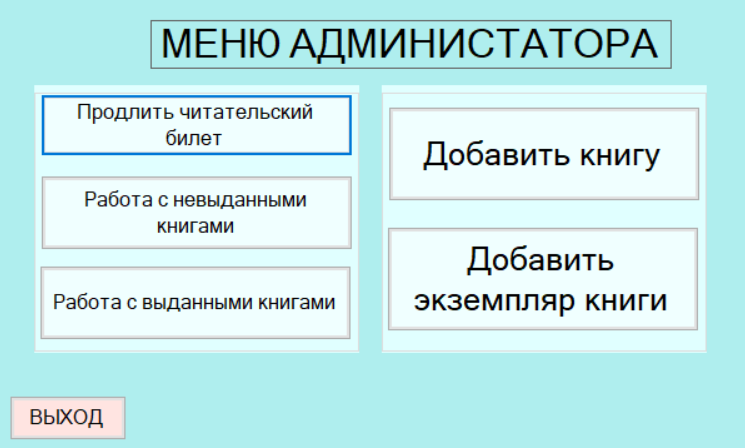
Войдя в приложение, пользователь может ознакомиться с книгами, которые он не забрал, дату, до которой они будут его ждать, а также с книгами, которые он забрал и дату, когда он должен их вернуть.



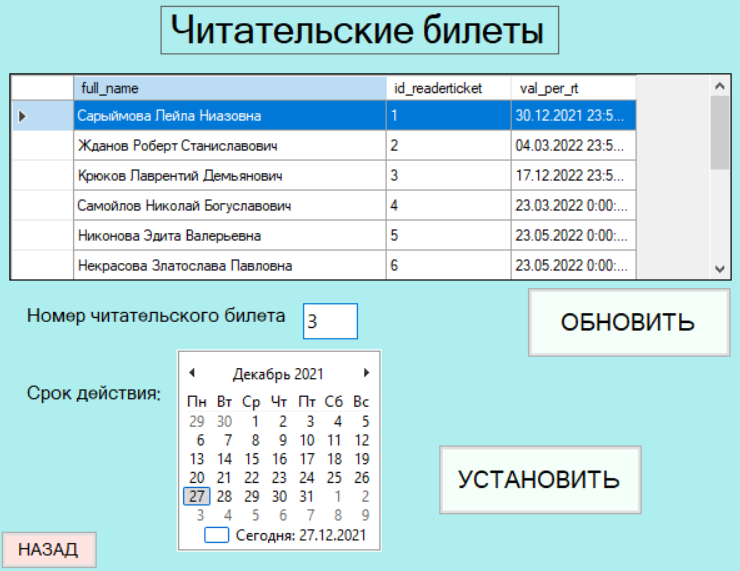
Нажав кнопку «Хочу взять книгу!», пользователь переходит в интерфейс выбора книги. Однако если его читательский билет просрочен, то сделать этого он не сможет. Здесь есть возможность посмотреть книги из конкретной библиотеки или найти нужную книгу и посмотреть в какой библиотеке она есть. Выбрав книгу и библиотеку, книга добавляется в таблицу книг, ожидающих выдачу.



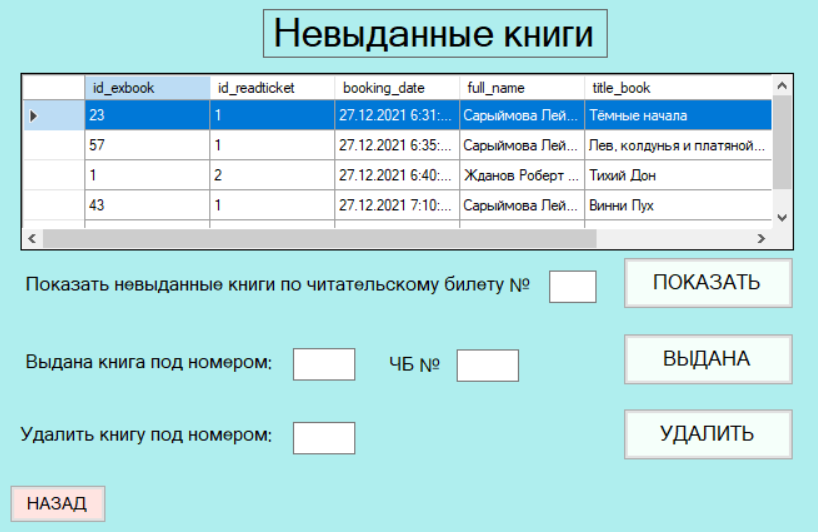
Если библиотекарь заходит в систему в качестве администратора, то ему открываются следующие возможности:



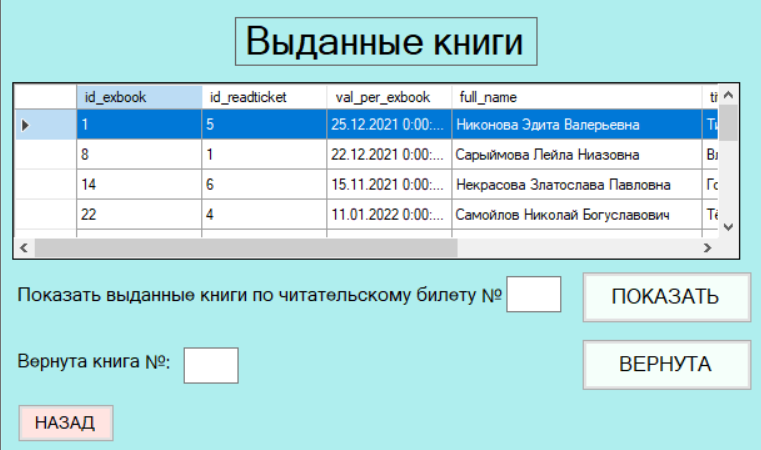
Продлить читательский билет можно, когда клиент пришел, оплатил услуги библиотеки на год и назвал номер своего читательского билета:



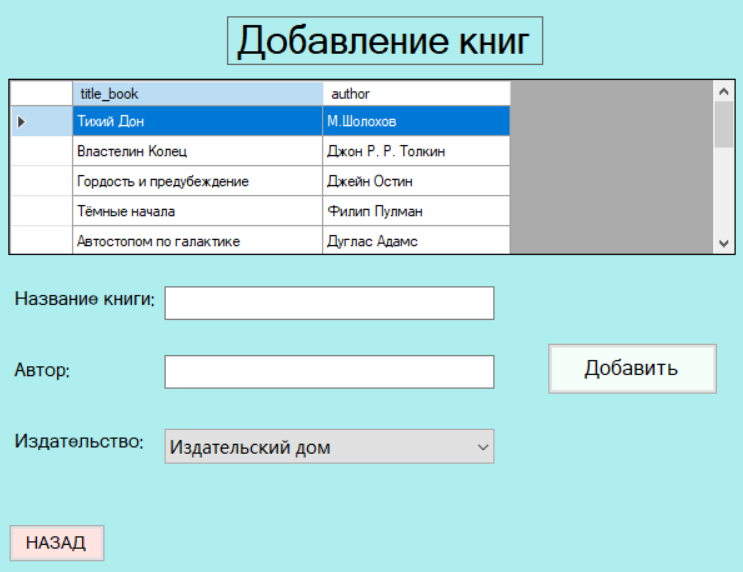
Следующая функция – работа с не выданными книгами. Когда клиент забронировал книгу, он приходит в библиотеку, чтобы забрать ее. Библиотекарь же выдает книгу и отмечает, что книга действительно выдана. Тогда книга отправляется в таблицу с выданными книгами, где срок ее возвращения обозначается как сегодняшняя дата + 14 дней. Если книга давно выдана, а забронировавший не пришел за ней, то библиотекарь может удалить книгу из этой таблицы.



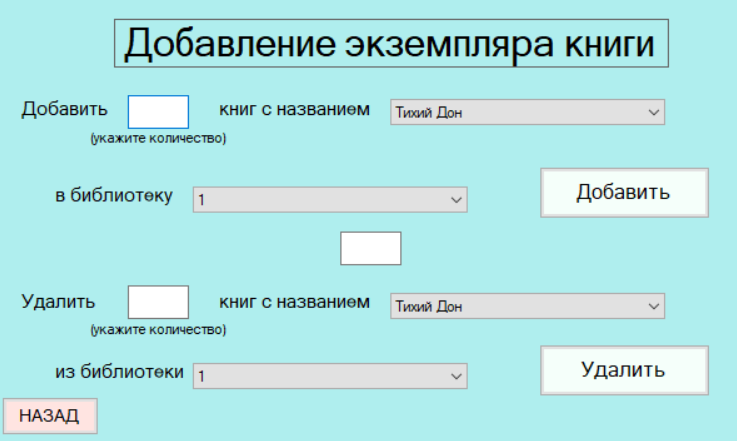
И, соответственно, работа с выданными книгами. Тут библиотекарь смотрит по книге номер экземпляра и вводит в графу. После этого книга удаляется из таблицы выданных книг, а в таблице экземпляров номер читательского билета устанавливается как null.



Если поступили новые книги, то сначала нужно добавить ее название, автора и издательства в таблицу с книгами.



А уже потом добавить экземпляры книг. Либо удалить какое-то количество списанных книг.



После каждой успешной операции всплывает окошко об успешном изменении данных.

# **Приложение 1. Сценарий создания базы данных.**

--

-- PostgreSQL database dump

--

-- Dumped from database version 13.4

-- Dumped by pg\_dump version 13.4

-- Started on 2021-12-27 02:43:52

SET statement\_timeout = 0;

SET lock\_timeout = 0;

SET idle\_in\_transaction\_session\_timeout = 0;

SET client\_encoding = 'UTF8';

SET standard\_conforming\_strings = on;

SELECT pg\_catalog.set\_config('search\_path', '', false);

SET check\_function\_bodies = false;

SET xmloption = content;

SET client\_min\_messages = warning;

SET row\_security = off;

SET default\_tablespace = '';

SET default\_table\_access\_method = heap;

--

-- TOC entry 201 (class 1259 OID 16397)

-- Name: accounts; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.accounts (

id\_acc integer NOT NULL,

full\_name character varying NOT NULL,

id\_readerticket integer NOT NULL,

login character varying NOT NULL,

password character varying NOT NULL,

phone\_acc character varying(10) NOT NULL,

birth\_date character varying NOT NULL

);

ALTER TABLE public.accounts OWNER TO postgres;

--

-- TOC entry 200 (class 1259 OID 16395)

-- Name: Accounts\_id\_acc\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

ALTER TABLE public.accounts ALTER COLUMN id\_acc ADD GENERATED ALWAYS AS IDENTITY (

SEQUENCE NAME public."Accounts\_id\_acc\_seq"

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1

);

--

-- TOC entry 202 (class 1259 OID 16405)

-- Name: books; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.books (

id\_book integer NOT NULL,

title\_book character varying NOT NULL,

author character varying NOT NULL,

id\_publisher integer NOT NULL

);

ALTER TABLE public.books OWNER TO postgres;

--

-- TOC entry 209 (class 1259 OID 16529)

-- Name: Books\_id\_book\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

ALTER TABLE public.books ALTER COLUMN id\_book ADD GENERATED ALWAYS AS IDENTITY (

SEQUENCE NAME public."Books\_id\_book\_seq"

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1

);

--

-- TOC entry 204 (class 1259 OID 16418)

-- Name: exbooks; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.exbooks (

id\_readticket integer,

id\_book integer NOT NULL,

id\_library integer NOT NULL,

id\_exbook integer NOT NULL

);

ALTER TABLE public.exbooks OWNER TO postgres;

--

-- TOC entry 208 (class 1259 OID 16527)

-- Name: Exbooks\_id\_exbook\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

ALTER TABLE public.exbooks ALTER COLUMN id\_exbook ADD GENERATED ALWAYS AS IDENTITY (

SEQUENCE NAME public."Exbooks\_id\_exbook\_seq"

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1

);

--

-- TOC entry 206 (class 1259 OID 16465)

-- Name: libraries; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.libraries (

id\_library integer NOT NULL,

address\_lib character varying NOT NULL,

phone\_lib character varying(11) NOT NULL

);

ALTER TABLE public.libraries OWNER TO postgres;

--

-- TOC entry 207 (class 1259 OID 16525)

-- Name: Libraries\_id\_library\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

ALTER TABLE public.libraries ALTER COLUMN id\_library ADD GENERATED ALWAYS AS IDENTITY (

SEQUENCE NAME public."Libraries\_id\_library\_seq"

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1

);

--

-- TOC entry 205 (class 1259 OID 16437)

-- Name: publishers; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.publishers (

id\_publisher integer NOT NULL,

title\_pub character varying NOT NULL,

address\_pub character varying NOT NULL,

phone\_pub character varying(11) NOT NULL

);

ALTER TABLE public.publishers OWNER TO postgres;

--

-- TOC entry 210 (class 1259 OID 16531)

-- Name: Publishers\_id\_publisher\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

ALTER TABLE public.publishers ALTER COLUMN id\_publisher ADD GENERATED ALWAYS AS IDENTITY (

SEQUENCE NAME public."Publishers\_id\_publisher\_seq"

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1

);

--

-- TOC entry 203 (class 1259 OID 16413)

-- Name: reader\_tickets; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.reader\_tickets (

id\_readerticket integer NOT NULL,

val\_per\_rt character varying NOT NULL

);

ALTER TABLE public.reader\_tickets OWNER TO postgres;

--

-- TOC entry 211 (class 1259 OID 16533)

-- Name: Reader\_tickets\_id\_readerticket\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

ALTER TABLE public.reader\_tickets ALTER COLUMN id\_readerticket ADD GENERATED ALWAYS AS IDENTITY (

SEQUENCE NAME public."Reader\_tickets\_id\_readerticket\_seq"

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1

);

--

-- TOC entry 213 (class 1259 OID 16581)

-- Name: not\_taked\_books; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.not\_taked\_books (

id\_exbook integer NOT NULL,

id\_readticket integer NOT NULL,

booking\_date character varying NOT NULL

);

ALTER TABLE public.not\_taked\_books OWNER TO postgres;

--

-- TOC entry 212 (class 1259 OID 16535)

-- Name: taked\_books; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.taked\_books (

id\_exbook integer NOT NULL,

id\_readticket integer NOT NULL,

val\_per\_exbook character varying

);

ALTER TABLE public.taked\_books OWNER TO postgres;

--

-- TOC entry 3047 (class 0 OID 16397)

-- Dependencies: 201

-- Data for Name: accounts; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.accounts (id\_acc, full\_name, id\_readerticket, login, password, phone\_acc, birth\_date) FROM stdin;

1 Сарыймова Лейла Ниазовна 1 saryymova@mail.ru 12341234 9600780854 13.03.2001 0:00:00

3 Жданов Роберт Станиславович 2 jjdanov@mail.ru 12341234 9612582475 12.07.2000 0:00:00

4 Крюков Лаврентий Демьянович 3 kryukov@mail.ru 12341234 9456317645 14.05.1965 0:00:00

5 Самойлов Николай Богуславович 4 samoilov@yandex.ru 12341234 9274398002 26.09.1976 0:00:00

6 Никонова Эдита Валерьевна 5 nikonova@mail.ru 12341234 9551274092 06.12.2005 0:00:00

7 Некрасова Златослава Павловна 6 nekrasova@mail.ru 12341234 9214606144 08.10.1990 0:00:00

8 Капустина Мелиана Руслановна 7 kapustin@mail.ru 12341234 9542157036 08.08.1995 0:00:00

9 Корнилова Артемида Демьяновна 8 kornilova@mail.ru 12341234 9221540975 05.06.2006 0:00:00

10 Шарапов Юлий Андреевич 9 sharapov@yandex.ru 12341234 9246530857 08.11.2001 0:00:00

11 Прохоров Самуил Денисович 10 prohorov@mail.ru 12341234 9552389942 31.03.1970 0:00:00

12 Гурьев Владислав Богуславович 11 guryev@mail.ru 12341234 9234502911 07.08.1999 0:00:00

13 Потапов Гордей Валерьевич 12 potapov@mail.ru 12341234 9653489600 23.04.1986 0:00:00

14 Бородавкина Ксения Маратовна 21 borodavkina@mail.ru 12341234 9511594556 13.06.2001 23:59:59

\.

--

-- TOC entry 3048 (class 0 OID 16405)

-- Dependencies: 202

-- Data for Name: books; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.books (id\_book, title\_book, author, id\_publisher) FROM stdin;

1 Тихий Дон М.Шолохов 1

2 Властелин Колец Джон Р. Р. Толкин\n\n 1

3 Гордость и предубеждение Джейн Остин 3

4 Тёмные начала Филип Пулман 2

5 Автостопом по галактике Дуглас Адамс 3

6 Гарри Поттер и Кубок огня Джоан Роулинг 3

7 Убить пересмешника Харпер Ли 1

9 1984 Джордж Оруэлл 3

10 Лев, колдунья и платяной шкаф Клайв Стэйплз Льюис 2

11 Скотный Двор Джордж Оруэлл 3

8 Винни Пух Алан Александр Милн 1

12 451 градус по Фаренгейту Рэй Бредбери 2

14 Однажды в Голливуде Тарантино К. 2

16 Колобок Народная 1

\.

--

-- TOC entry 3050 (class 0 OID 16418)

-- Dependencies: 204

-- Data for Name: exbooks; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.exbooks (id\_readticket, id\_book, id\_library, id\_exbook) FROM stdin;

1 1 2 2

\N 1 1 4

\N 1 3 5

2 1 3 6

\N 2 1 7

\N 2 2 9

\N 2 2 10

3 2 3 11

\N 2 3 12

\N 3 1 13

\N 3 2 15

2 3 2 16

\N 3 3 17

\N 3 3 18

1 2 2 8

\N 4 3 24

\N 5 1 26

\N 5 2 27

\N 5 2 28

\N 5 3 29

\N 6 1 31

\N 6 1 32

\N 6 2 34

\N 6 3 35

\N 6 3 36

\N 7 1 38

\N 7 2 39

\N 7 2 40

\N 7 3 41

\N 7 3 42

\N 8 1 44

\N 8 3 47

\N 8 3 48

\N 9 1 49

\N 9 1 50

\N 9 2 51

\N 9 2 52

\N 9 3 53

\N 9 3 54

\N 10 1 55

\N 10 1 56

\N 10 2 58

\N 10 3 59

\N 4 2 21

4 4 2 22

6 3 1 14

7 5 1 25

8 5 3 30

9 6 2 33

10 7 1 37

12 10 3 60

\N 10 3 61

\N 10 3 62

\N 8 2 46

\N 1 1 63

\N 1 1 64

\N 1 1 65

\N 1 1 66

1 4 1 19

1 4 1 20

1 1 1 3

1 4 3 23

1 10 2 57

2 1 2 1

1 8 1 43

1 8 2 45

\N 11 3 67

\N 11 3 68

\.

--

-- TOC entry 3052 (class 0 OID 16465)

-- Dependencies: 206

-- Data for Name: libraries; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.libraries (id\_library, address\_lib, phone\_lib) FROM stdin;

1 Казань, ул. Пушкина, д. 86 88432228284

2 Казань, ул.Карла Маркса, д. 36 88432926826

3 Казань, ул. Кремлевская, д. 35 88432337180

\.

--

-- TOC entry 3059 (class 0 OID 16581)

-- Dependencies: 213

-- Data for Name: not\_taked\_books; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.not\_taked\_books (id\_exbook, id\_readticket, booking\_date) FROM stdin;

23 1 27.12.2021 6:31:38

57 1 27.12.2021 6:35:55

1 2 27.12.2021 6:40:59

43 1 27.12.2021 7:10:21

45 1 27.12.2021 18:12:01

\.

--

-- TOC entry 3051 (class 0 OID 16437)

-- Dependencies: 205

-- Data for Name: publishers; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.publishers (id\_publisher, title\_pub, address\_pub, phone\_pub) FROM stdin;

1 Издательский дом Казань, ул. Парижской Коммуны, д. 10 89872642922

2 Издательство Сергея Бузукина Казань, ул. ​Мусы Джалиля, д. 7 89172979875

3 Ак Буре Казань, ул. ​Декабристов, 2 89872906155

\.

--

-- TOC entry 3049 (class 0 OID 16413)

-- Dependencies: 203

-- Data for Name: reader\_tickets; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.reader\_tickets (id\_readerticket, val\_per\_rt) FROM stdin;

22 17.02.2022 18:07:25

23 17.02.2022 18:10:23

3 17.12.2022 23:59:59

4 23.03.2022 0:00:00

5 23.05.2022 0:00:00

6 23.05.2022 0:00:00

7 23.05.2022 0:00:00

8 23.05.2022 0:00:00

9 4.08.2022 0:00:00

10 4.08.2022 0:00:00

11 4.12.2021 0:00:00

12 4.08.2022 0:00:00

21 4.08.2022 0:00:00

1 30.12.2021 23:59:59

2 04.03.2022 23:59:59

\.

--

-- TOC entry 3058 (class 0 OID 16535)

-- Dependencies: 212

-- Data for Name: taked\_books; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.taked\_books (id\_exbook, id\_readticket, val\_per\_exbook) FROM stdin;

1 5 25.12.2021 0:00:00

8 1 22.12.2021 0:00:00

14 6 15.11.2021 0:00:00

22 4 11.01.2022 0:00:00

25 7 21.01.2022 0:00:00

30 8 15.01.2022 0:00:00

33 9 07.12.2021 0:00:00

37 10 17.12.2021 0:00:00

60 12 31.12.2021 0:00:00

11 3 15.11.2021 0:00:00

16 2 30.12.2021 7:06:06

6 2 31.12.2021 7:08:12

16 2 31.12.2021 18:13:25

\.

--

-- TOC entry 3065 (class 0 OID 0)

-- Dependencies: 200

-- Name: Accounts\_id\_acc\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public."Accounts\_id\_acc\_seq"', 14, true);

--

-- TOC entry 3066 (class 0 OID 0)

-- Dependencies: 209

-- Name: Books\_id\_book\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public."Books\_id\_book\_seq"', 16, true);

--

-- TOC entry 3067 (class 0 OID 0)

-- Dependencies: 208

-- Name: Exbooks\_id\_exbook\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public."Exbooks\_id\_exbook\_seq"', 68, true);

--

-- TOC entry 3068 (class 0 OID 0)

-- Dependencies: 207

-- Name: Libraries\_id\_library\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public."Libraries\_id\_library\_seq"', 3, true);

--

-- TOC entry 3069 (class 0 OID 0)

-- Dependencies: 210

-- Name: Publishers\_id\_publisher\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public."Publishers\_id\_publisher\_seq"', 3, true);

--

-- TOC entry 3070 (class 0 OID 0)

-- Dependencies: 211

-- Name: Reader\_tickets\_id\_readerticket\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public."Reader\_tickets\_id\_readerticket\_seq"', 23, true);

--

-- TOC entry 2896 (class 2606 OID 16404)

-- Name: accounts Accounts\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.accounts

ADD CONSTRAINT "Accounts\_pkey" PRIMARY KEY (id\_acc);

--

-- TOC entry 2898 (class 2606 OID 16412)

-- Name: books Books\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.books

ADD CONSTRAINT "Books\_pkey" PRIMARY KEY (id\_book);

--

-- TOC entry 2902 (class 2606 OID 16425)

-- Name: exbooks Exbooks\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.exbooks

ADD CONSTRAINT "Exbooks\_pkey" PRIMARY KEY (id\_exbook);

--

-- TOC entry 2906 (class 2606 OID 16472)

-- Name: libraries Libraries\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.libraries

ADD CONSTRAINT "Libraries\_pkey" PRIMARY KEY (id\_library);

--

-- TOC entry 2904 (class 2606 OID 16444)

-- Name: publishers Publishers\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.publishers

ADD CONSTRAINT "Publishers\_pkey" PRIMARY KEY (id\_publisher);

--

-- TOC entry 2900 (class 2606 OID 16417)

-- Name: reader\_tickets Reader\_tickets\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.reader\_tickets

ADD CONSTRAINT "Reader\_tickets\_pkey" PRIMARY KEY (id\_readerticket);

--

-- TOC entry 2907 (class 2606 OID 16445)

-- Name: accounts accounts\_readtickets; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.accounts

ADD CONSTRAINT accounts\_readtickets FOREIGN KEY (id\_readerticket) REFERENCES public.reader\_tickets(id\_readerticket) NOT VALID;

--

-- TOC entry 2908 (class 2606 OID 16450)

-- Name: books books\_publishers; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.books

ADD CONSTRAINT books\_publishers FOREIGN KEY (id\_publisher) REFERENCES public.publishers(id\_publisher) NOT VALID;

--

-- TOC entry 2910 (class 2606 OID 16488)

-- Name: exbooks exbooks\_books; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.exbooks

ADD CONSTRAINT exbooks\_books FOREIGN KEY (id\_book) REFERENCES public.books(id\_book) NOT VALID;

--

-- TOC entry 2911 (class 2606 OID 16538)

-- Name: exbooks exbooks\_libraries; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.exbooks

ADD CONSTRAINT exbooks\_libraries FOREIGN KEY (id\_library) REFERENCES public.libraries(id\_library) NOT VALID;

--

-- TOC entry 2909 (class 2606 OID 16483)

-- Name: exbooks exbooks\_readtickets; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.exbooks

ADD CONSTRAINT exbooks\_readtickets FOREIGN KEY (id\_readticket) REFERENCES public.reader\_tickets(id\_readerticket) NOT VALID;

--

-- TOC entry 2914 (class 2606 OID 16587)

-- Name: not\_taked\_books ntb\_exbook; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.not\_taked\_books

ADD CONSTRAINT ntb\_exbook FOREIGN KEY (id\_exbook) REFERENCES public.exbooks(id\_exbook) NOT VALID;

--

-- TOC entry 2915 (class 2606 OID 16592)

-- Name: not\_taked\_books ntb\_readticket; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.not\_taked\_books

ADD CONSTRAINT ntb\_readticket FOREIGN KEY (id\_readticket) REFERENCES public.reader\_tickets(id\_readerticket) NOT VALID;

--

-- TOC entry 2912 (class 2606 OID 16548)

-- Name: taked\_books takbooks\_exbooks; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.taked\_books

ADD CONSTRAINT takbooks\_exbooks FOREIGN KEY (id\_exbook) REFERENCES public.exbooks(id\_exbook) NOT VALID;

--

-- TOC entry 2913 (class 2606 OID 16553)

-- Name: taked\_books takbooks\_readtickets; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.taked\_books

ADD CONSTRAINT takbooks\_readtickets FOREIGN KEY (id\_readticket) REFERENCES public.reader\_tickets(id\_readerticket) NOT VALID;

-- Completed on 2021-12-27 02:43:53

--

-- PostgreSQL database dump complete

--

# **Приложение 2 Программный код клиентского приложения**

**AddBook.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Npgsql;

using NpgsqlTypes;

namespace OnlineLibrary

{

public partial class AddBook : Form

{

public AddBook()

{

InitializeComponent();

comboBox1.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;

comboBox1.SelectedIndex = 0;

}

DataSet ds;

DB db;

NpgsqlDataAdapter adapter;

private void AddBook\_Load(object sender, EventArgs e)

{

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

db = new DB();

ds = new DataSet();

adapter = new NpgsqlDataAdapter();

db.openConnection();

NpgsqlCommand command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT books.title\_book, books.author FROM books";

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[0].Width = 200;

dataGridView1.Columns[1].Width = 150;

}

private void button4\_Click(object sender, EventArgs e)

{

string title = textBox1.Text;

string author = textBox2.Text;

int i = comboBox1.SelectedIndex;

int publish = i + 1;

NpgsqlDataAdapter da = new NpgsqlDataAdapter("", db.getConnection());

da.InsertCommand = new NpgsqlCommand("INSERT INTO books (\"title\_book\", \"author\", \"id\_publisher\") values (:title\_book, :author, :id\_publisher)", db.getConnection());

da.InsertCommand.Parameters.Add(new NpgsqlParameter("title\_book", NpgsqlDbType.Varchar));

da.InsertCommand.Parameters.Add(new NpgsqlParameter("author", NpgsqlDbType.Varchar));

da.InsertCommand.Parameters.Add(new NpgsqlParameter("id\_publisher", NpgsqlDbType.Integer));

da.InsertCommand.Parameters[0].Direction = ParameterDirection.Input;

da.InsertCommand.Parameters[1].Direction = ParameterDirection.Input;

da.InsertCommand.Parameters[2].Direction = ParameterDirection.Input;

da.InsertCommand.Parameters[0].SourceColumn = "title\_book";

da.InsertCommand.Parameters[1].SourceColumn = "author";

da.InsertCommand.Parameters[2].SourceColumn = "id\_publisher";

da.InsertCommand.Parameters[0].Value = title;

da.InsertCommand.Parameters[1].Value = author;

da.InsertCommand.Parameters[2].Value = publish;

da.InsertCommand.ExecuteNonQuery();

MessageBox.Show("Книга добавлена!");

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

db = new DB();

ds = new DataSet();

adapter = new NpgsqlDataAdapter();

db.openConnection();

NpgsqlCommand command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT books.title\_book, books.author FROM books";

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[0].Width = 200;

dataGridView1.Columns[1].Width = 150;

}

private void button3\_Click(object sender, EventArgs e)

{

this.Hide();

AdminForm mainForm = new AdminForm();

mainForm.Show();

}

}

}

AddExbook.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Npgsql;

using NpgsqlTypes;

namespace OnlineLibrary

{

public partial class AddExbook : Form

{

public AddExbook()

{

InitializeComponent();

comboBox2.SelectedIndex = 0;

comboBox4.SelectedIndex = 0;

}

DataSet ds;

DB db;

NpgsqlDataAdapter adapter;

private void AddExbook\_Load(object sender, EventArgs e)

{

DB db = new DB();

DataTable dt = new DataTable();

NpgsqlCommand command = new NpgsqlCommand("SELECT books.title\_book FROM books", db.getConnection());

NpgsqlDataAdapter adapter = new NpgsqlDataAdapter("", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(dt);

if (dt.Rows.Count > 0)

{

for (int i=0; i<dt.Rows.Count; i++)

{

DataRow row = dt.Rows[i];

string titles = row["title\_book"].ToString();

comboBox1.Items.Add(titles);

comboBox3.Items.Add(titles);

comboBox1.SelectedIndex = 0;

comboBox3.SelectedIndex = 0;

}

}

comboBox1.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;

comboBox2.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;

comboBox3.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;

comboBox4.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;

}

private void button4\_Click(object sender, EventArgs e)

{

int count = int.Parse(textBox1.Text);

int libr = comboBox2.SelectedIndex + 1;

string book\_title = comboBox1.SelectedItem.ToString();

//идeнтификатор книги ищем

int id\_book = 0;

DB db = new DB();

DataTable dt = new DataTable();

dt = new DataTable();

NpgsqlCommand command = new NpgsqlCommand("SELECT books.id\_book FROM books WHERE title\_book = (@title\_book)", db.getConnection());

command.Parameters.Add("@title\_book", NpgsqlDbType.Varchar).Value = book\_title;

adapter = new NpgsqlDataAdapter("", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(dt);

if (dt.Rows.Count > 0)

{

DataRow row = dt.Rows[0];

string titles = row["id\_book"].ToString();

id\_book = int.Parse(titles);

}

for (int i = 0; i < count; i++)

{

db.openConnection();

NpgsqlDataAdapter da = new NpgsqlDataAdapter("", db.getConnection());

da.InsertCommand = new NpgsqlCommand("INSERT INTO exbooks (\"id\_book\", \"id\_library\") values(:id\_book, :id\_library)", db.getConnection());

da.InsertCommand.Parameters.Add(new NpgsqlParameter("id\_book", NpgsqlDbType.Integer));

da.InsertCommand.Parameters.Add(new NpgsqlParameter("id\_library", NpgsqlDbType.Integer));

da.InsertCommand.Parameters[0].Direction = ParameterDirection.Input;

da.InsertCommand.Parameters[1].Direction = ParameterDirection.Input;

da.InsertCommand.Parameters[0].SourceColumn = "id\_book";

da.InsertCommand.Parameters[1].SourceColumn = "id\_library";

da.InsertCommand.Parameters[0].Value = id\_book;

da.InsertCommand.Parameters[1].Value = libr;

da.InsertCommand.ExecuteNonQuery();

}

MessageBox.Show("Книга(и) добавлена(ы)!");

}

private void button1\_Click(object sender, EventArgs e)

{

int count = int.Parse(textBox2.Text);

int libr = comboBox2.SelectedIndex + 1;

string book\_title = comboBox3.SelectedItem.ToString();

//идeнтификатор книги ищем

int id\_book = 0;

DB db = new DB();

DataTable dt = new DataTable();

dt = new DataTable();

NpgsqlCommand command = new NpgsqlCommand("SELECT books.id\_book FROM books WHERE title\_book = (@title\_book)", db.getConnection());

command.Parameters.Add("@title\_book", NpgsqlDbType.Varchar).Value = book\_title;

adapter = new NpgsqlDataAdapter("", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(dt);

if (dt.Rows.Count > 0)

{

DataRow row = dt.Rows[0];

string titles = row["id\_book"].ToString();

id\_book = int.Parse(titles);

}

for (int i = 0; i < count; i++)

{

dt = new DataTable();

command = new NpgsqlCommand("SELECT exbooks.id\_exbook FROM exbooks WHERE id\_book = (@id\_book) AND id\_library = (@id\_library)", db.getConnection());

command.Parameters.Add("@id\_book", NpgsqlDbType.Integer).Value = id\_book;

command.Parameters.Add("@id\_library", NpgsqlDbType.Integer).Value = libr;

adapter = new NpgsqlDataAdapter("", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(dt);

if (dt.Rows.Count > 0)

{

DataRow row = dt.Rows[0];

int id\_exbook = int.Parse(row["id\_exbook"].ToString());

textBox3.Text = id\_exbook.ToString();

db = new DB();

NpgsqlCommand ccommand = new NpgsqlCommand("DELETE FROM exbooks WHERE id\_exbook = @id\_exbook AND id\_readticket IS NULL", db.getConnection());

ccommand.Parameters.Add("@id\_exbook", NpgsqlDbType.Integer).Value = id\_exbook;

db.openConnection();

}

MessageBox.Show("Книга(и) удалена(и)!");

}

}

private void button3\_Click(object sender, EventArgs e)

{

this.Hide();

AdminForm mainForm = new AdminForm();

mainForm.Show();

}

private void keypress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

}

}

}

}

AdminForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace OnlineLibrary

{

public partial class AdminForm : Form

{

public AdminForm()

{

InitializeComponent();

}

private void AdminForm\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

this.Hide();

Readertickets mainForm = new Readertickets();

mainForm.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

this.Hide();

ntbForm mainForm = new ntbForm();

mainForm.Show();

}

private void button3\_Click(object sender, EventArgs e)

{

this.Hide();

tbForm mainForm = new tbForm();

mainForm.Show();

}

private void button4\_Click(object sender, EventArgs e)

{

this.Hide();

AddBook mainForm = new AddBook();

mainForm.Show();

}

private void button5\_Click(object sender, EventArgs e)

{

this.Hide();

AddExbook mainForm = new AddExbook();

mainForm.Show();

}

private void button6\_Click(object sender, EventArgs e)

{

this.Hide();

Autorisation mainForm = new Autorisation();

mainForm.Show();

}

}

}

**Autorization.cs**

using Npgsql;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using NpgsqlTypes;

namespace OnlineLibrary

{

public partial class Autorisation : Form

{

private const string adminLogin = "admin";

private const string adminPassword = "admin";

public Autorisation()

{

InitializeComponent();

Font font = new Font("Segoe UI Semilight", 10);

label1.Font = font;

label2.Font = font;

label3.Font = font;

login.Font = font;

password.Font = font;

label4.Font = font;

button1.Font = font;

button2.Font = font;

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

this.Hide();

Registration mainForm = new Registration();

mainForm.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

String loginUser = login.Text;

String passwUser = password.Text;

DB db = new DB();

db.openConnection();

DataTable table = new DataTable();

NpgsqlDataAdapter adapter = new NpgsqlDataAdapter();

NpgsqlCommand command = new NpgsqlCommand("SELECT \* FROM accounts WHERE login = @lU AND password = @pU", db.getConnection());

command.Parameters.Add("@lU", NpgsqlDbType.Varchar).Value = loginUser;

command.Parameters.Add("@pU", NpgsqlDbType.Varchar).Value = passwUser;

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

this.Hide();

MessageBox.Show("Успешно!");

UserForm userForm = new UserForm();

userForm.acc\_login = loginUser;

userForm.Show();

}

else if (login.Text == adminLogin && password.Text == adminPassword)

{

this.Hide();

MessageBox.Show("Успешная авторизация!", "Админ-аккаунт", MessageBoxButtons.OK, MessageBoxIcon.Information);

AdminForm mainForm = new AdminForm();

mainForm.Show();

}

else

MessageBox.Show("Неверный логин или пароль", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

**DB.cs**

using Npgsql;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OnlineLibrary

{

class DB

{

NpgsqlConnection connection = new NpgsqlConnection("Server = localhost; Port = 5432; User Id = postgres; Password = postgres; Database = my\_bd;");

public void openConnection()

{

if (connection.State == System.Data.ConnectionState.Closed)

{

connection.Open();

}

}

//закрываем соединение с бд

public void closeConnection()

{

if (connection.State == System.Data.ConnectionState.Open)

{

connection.Close();

}

}

// соединение с бд

public NpgsqlConnection getConnection()

{

return connection;

}

}

}

**ntbForm.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Npgsql;

using NpgsqlTypes;

namespace OnlineLibrary

{

public partial class ntbForm : Form

{

DataSet ds;

DB db;

NpgsqlDataAdapter adapter;

public ntbForm()

{

InitializeComponent();

}

private void ntbForm\_Load(object sender, EventArgs e)

{

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

db = new DB();

ds = new DataSet();

adapter = new NpgsqlDataAdapter();

db.openConnection();

NpgsqlCommand command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT not\_taked\_books.id\_exbook, not\_taked\_books.id\_readticket, not\_taked\_books.booking\_date, accounts.full\_name, books.title\_book, books.author FROM not\_taked\_books INNER JOIN accounts ON accounts.id\_readerticket = not\_taked\_books.id\_readticket JOIN exbooks ON exbooks.id\_exbook = not\_taked\_books.id\_exbook JOIN books ON books.id\_book = exbooks.id\_book";

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[4].Width = 150;

dataGridView1.Columns[5].Width = 150;

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

db = new DB();

ds = new DataSet();

adapter = new NpgsqlDataAdapter();

db.openConnection();

NpgsqlCommand command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT not\_taked\_books.id\_exbook, not\_taked\_books.id\_readticket, not\_taked\_books.booking\_date, accounts.full\_name, books.title\_book, books.author FROM not\_taked\_books INNER JOIN accounts ON accounts.id\_readerticket = not\_taked\_books.id\_readticket JOIN exbooks ON exbooks.id\_exbook = not\_taked\_books.id\_exbook JOIN books ON books.id\_book = exbooks.id\_book WHERE not\_taked\_books.id\_readticket = @id\_readticket";

command.Parameters.Add("@id\_readticket", NpgsqlDbType.Integer).Value = int.Parse(textBox1.Text);

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[4].Width = 150;

dataGridView1.Columns[5].Width = 150;

}

private void button1\_Click(object sender, EventArgs e)

{

int id\_exbook = int.Parse(textBox2.Text);

int id\_rt = int.Parse(textBox3.Text);

NpgsqlCommand command = new NpgsqlCommand("DELETE FROM not\_taked\_books WHERE id\_exbook = @id\_exbook", db.getConnection());

DateTime now = DateTime.Now.AddDays(14);

command.Parameters.Add("@id\_exbook", NpgsqlDbType.Integer).Value = id\_exbook;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

db = new DB();

ds = new DataSet();

adapter = new NpgsqlDataAdapter();

db.openConnection();

command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT not\_taked\_books.id\_exbook, not\_taked\_books.id\_readticket, not\_taked\_books.booking\_date, accounts.full\_name, books.title\_book, books.author FROM not\_taked\_books INNER JOIN accounts ON accounts.id\_readerticket = not\_taked\_books.id\_readticket JOIN exbooks ON exbooks.id\_exbook = not\_taked\_books.id\_exbook JOIN books ON books.id\_book = exbooks.id\_book WHERE not\_taked\_books.id\_readticket = @id\_readticket";

command.Parameters.Add("@id\_readticket", NpgsqlDbType.Integer).Value = int.Parse(textBox3.Text);

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[4].Width = 150;

dataGridView1.Columns[5].Width = 150;

command = new NpgsqlCommand("INSERT INTO taked\_books (id\_exbook, id\_readticket, val\_per\_exbook) VALUES (@id\_exbook, @id\_readticket, @val\_per\_exbook)", db.getConnection());

command.Parameters.Add("@id\_exbook", NpgsqlDbType.Integer).Value = id\_exbook;

command.Parameters.Add("@id\_readticket", NpgsqlDbType.Integer).Value = id\_rt;

command.Parameters.Add("@val\_per\_exbook", NpgsqlDbType.Varchar).Value = now.ToString();

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Книга выдана!");

}

}

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void button4\_Click(object sender, EventArgs e)

{

NpgsqlCommand command = new NpgsqlCommand("DELETE FROM not\_taked\_books WHERE id\_exbook = @id\_exbook", db.getConnection());

DateTime now = DateTime.Now.AddDays(14);

command.Parameters.Add("@id\_exbook", NpgsqlDbType.Integer).Value = int.Parse(textBox4.Text);

db.openConnection();

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

db = new DB();

ds = new DataSet();

adapter = new NpgsqlDataAdapter();

db.openConnection();

command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT not\_taked\_books.id\_exbook, not\_taked\_books.id\_readticket, not\_taked\_books.booking\_date, accounts.full\_name, books.title\_book, books.author FROM not\_taked\_books INNER JOIN accounts ON accounts.id\_readerticket = not\_taked\_books.id\_readticket JOIN exbooks ON exbooks.id\_exbook = not\_taked\_books.id\_exbook JOIN books ON books.id\_book = exbooks.id\_book WHERE not\_taked\_books.id\_readticket = @id\_readticket";

command.Parameters.Add("@id\_readticket", NpgsqlDbType.Integer).Value = int.Parse(textBox4.Text);

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[4].Width = 150;

dataGridView1.Columns[5].Width = 150;

if (command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Книга удалена из таблицы невыданных книг");

}

}

private void button3\_Click(object sender, EventArgs e)

{

this.Hide();

AdminForm mainForm = new AdminForm();

mainForm.Show();

}

private void keypress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

}

}

}

}

**Readerickets.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Npgsql;

using NpgsqlTypes;

namespace OnlineLibrary

{

public partial class Readertickets : Form

{

DataSet ds;

DB db;

NpgsqlDataAdapter adapter;

public Readertickets()

{

InitializeComponent();

}

private void Readertickets\_Load\_1(object sender, EventArgs e)

{

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

db = new DB();

ds = new DataSet();

adapter = new NpgsqlDataAdapter();

db.openConnection();

NpgsqlCommand command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT accounts.full\_name, reader\_tickets.id\_readerticket, reader\_tickets.val\_per\_rt FROM accounts INNER JOIN reader\_tickets ON accounts.id\_readerticket = reader\_tickets.id\_readerticket";

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[0].Width = 250;

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

DB db = new DB();

DataTable table = new DataTable();

NpgsqlDataAdapter adapter = new NpgsqlDataAdapter();

NpgsqlCommand command = new NpgsqlCommand("UPDATE reader\_tickets SET val\_per\_rt = @val\_per\_rt WHERE id\_readerticket = @id\_readerticket", db.getConnection());

command.Parameters.Add("@val\_per\_rt", NpgsqlDbType.Varchar).Value = monthCalendar1.SelectionEnd.ToString();

command.Parameters.Add("@id\_readerticket", NpgsqlDbType.Integer).Value = int.Parse(textBox1.Text);

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count == 0)

{

MessageBox.Show("Изменения сохранены!");

}

}

private void button2\_Click(object sender, EventArgs e)

{

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

db = new DB();

ds = new DataSet();

adapter = new NpgsqlDataAdapter();

db.openConnection();

NpgsqlCommand command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT accounts.full\_name, reader\_tickets.id\_readerticket, reader\_tickets.val\_per\_rt FROM accounts INNER JOIN reader\_tickets ON accounts.id\_readerticket = reader\_tickets.id\_readerticket";

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[0].Width = 250;

}

private void button3\_Click(object sender, EventArgs e)

{

this.Hide();

AdminForm mainForm = new AdminForm();

mainForm.Show();

}

private void keypress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

}

}

}

}

**Registration.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Numerics;

using NpgsqlTypes;

using Npgsql;

namespace OnlineLibrary

{

public partial class Registration : Form

{

public Registration()

{

InitializeComponent();

Font font = new Font("Segoe UI Semilight", 10);

label1.Font = font;

label2.Font = font;

label3.Font = font;

label4.Font = font;

label5.Font = font;

label6.Font = font;

label7.Font = font;

textBox1.Font = font;

textBox2.Font = font;

textBox3.Font = font;

textBox4.Font = font;

monthCalendar1.Font = font;

button1.Font = font;

button2.Font = font;

button3.Font = font;

}

private void Registration\_Load(object sender, EventArgs e)

{

}

private void phone\_keypress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

MessageBox.Show("Ошибка! Номер телефона не может содержать буквы!");

}

}

private void phone\_textchanged(object sender, EventArgs e)

{

if (textBox2.TextLength == 10)

{

textBox2.ForeColor = Color.Black;

}

else

{

textBox2.ForeColor = Color.Red;

}

}

private void btn\_md(object sender, MouseEventArgs e)

{

textBox4.UseSystemPasswordChar = false;

}

private void btn\_mu(object sender, MouseEventArgs e)

{

textBox4.UseSystemPasswordChar = true;

}

private void button2\_Click(object sender, EventArgs e)

{

string fullname = textBox1.Text;

DateTime birthdate = monthCalendar1.SelectionEnd;

string phone = textBox2.Text;

string mail = textBox3.Text;

string password = textBox4.Text;

int countWords = fullname.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries).Length;

if (countWords != 3)

{

MessageBox.Show("Введите ФИО правильно!");

return;

}

if(textBox2.TextLength != 10)

{

MessageBox.Show("Введите телефон правильно!");

return;

}

if(!mail.Contains('@') || !mail.Contains('.'))

{

MessageBox.Show("Введите email правильно!");

return;

}

if(ismailfree())

{

return;

}

if (isphonefree())

{

return;

}

DB db = new DB();

NpgsqlCommand command = new NpgsqlCommand("INSERT INTO reader\_tickets (val\_per\_rt) VALUES (@val\_per\_rt)", db.getConnection());

DateTime now = DateTime.Now;

command.Parameters.Add("@val\_per\_rt", NpgsqlDbType.Varchar).Value = now.AddMonths(2).ToString();

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

DataTable dt = new DataTable();

string tic = "";

command = new NpgsqlCommand("SELECT \* FROM reader\_tickets WHERE val\_per\_rt = (@val\_per\_rt)", db.getConnection());

command.Parameters.Add("@val\_per\_rt", NpgsqlDbType.Varchar).Value = now.ToString();

NpgsqlDataAdapter adapter = new NpgsqlDataAdapter("", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(dt);

if (dt.Rows.Count > 0)

{

DataRow row = dt.Rows[0];

tic = row["id\_readerticket"].ToString();

textBox1.Text = tic;

}

int ticc = int.Parse(tic);

NpgsqlDataAdapter da = new NpgsqlDataAdapter("", db.getConnection());

da.InsertCommand = new NpgsqlCommand("INSERT INTO accounts (\"full\_name\", \"id\_readerticket\", \"login\", \"password\", \"phone\_acc\", \"birth\_date\") values(:full\_name, :id\_readerticket, :login, :password, :phone\_acc, :birth\_date)", db.getConnection());

da.InsertCommand.Parameters.Add(new NpgsqlParameter("full\_name", NpgsqlDbType.Varchar));

da.InsertCommand.Parameters.Add(new NpgsqlParameter("id\_readerticket", NpgsqlDbType.Integer));

da.InsertCommand.Parameters.Add(new NpgsqlParameter("login", NpgsqlDbType.Varchar));

da.InsertCommand.Parameters.Add(new NpgsqlParameter("password", NpgsqlDbType.Varchar));

da.InsertCommand.Parameters.Add(new NpgsqlParameter("phone\_acc", NpgsqlDbType.Varchar));

da.InsertCommand.Parameters.Add(new NpgsqlParameter("birth\_date", NpgsqlDbType.Varchar));

da.InsertCommand.Parameters[0].Direction = ParameterDirection.Input;

da.InsertCommand.Parameters[1].Direction = ParameterDirection.Input;

da.InsertCommand.Parameters[2].Direction = ParameterDirection.Input;

da.InsertCommand.Parameters[3].Direction = ParameterDirection.Input;

da.InsertCommand.Parameters[4].Direction = ParameterDirection.Input;

da.InsertCommand.Parameters[5].Direction = ParameterDirection.Input;

da.InsertCommand.Parameters[0].SourceColumn = "full\_name";

da.InsertCommand.Parameters[1].SourceColumn = "id\_readerticket";

da.InsertCommand.Parameters[2].SourceColumn = "login";

da.InsertCommand.Parameters[3].SourceColumn = "password";

da.InsertCommand.Parameters[4].SourceColumn = "phone\_acc";

da.InsertCommand.Parameters[5].SourceColumn = "birth\_date";

da.InsertCommand.Parameters[0].Value = fullname;

da.InsertCommand.Parameters[1].Value = ticc;

da.InsertCommand.Parameters[2].Value = mail;

da.InsertCommand.Parameters[3].Value = password;

da.InsertCommand.Parameters[4].Value = phone;

da.InsertCommand.Parameters[5].Value = birthdate.ToString();

da.InsertCommand.ExecuteNonQuery();

MessageBox.Show("Аккаунт был создан");

}

else

MessageBox.Show("Аккаунт не был создан", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

db.closeConnection();

}

public Boolean ismailfree()

{

DB db = new DB();

DataTable table = new DataTable();

NpgsqlDataAdapter adapter = new NpgsqlDataAdapter();

NpgsqlCommand command = new NpgsqlCommand("SELECT \* FROM accounts WHERE login = @lU", db.getConnection());

command.Parameters.Add("@lU", NpgsqlDbType.Varchar).Value = textBox3.Text;

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

MessageBox.Show("Данный логин уже занят, введите другой", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return true;

}

else

return false;

}

public Boolean isphonefree()

{

DB db = new DB();

DataTable table = new DataTable();

NpgsqlDataAdapter adapter = new NpgsqlDataAdapter();

NpgsqlCommand command = new NpgsqlCommand("SELECT \* FROM accounts WHERE phone\_acc = @phone", db.getConnection());

command.Parameters.Add("@phone", NpgsqlDbType.Varchar).Value = textBox2.Text;

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

MessageBox.Show("Данный телефон уже занят, введите другой", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return true;

}

else

return false;

}

private void button3\_Click(object sender, EventArgs e)

{

this.Hide();

Autorisation mainForm = new Autorisation();

mainForm.Show();

}

}

}

**tbForm.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Npgsql;

using NpgsqlTypes;

namespace OnlineLibrary

{

public partial class tbForm : Form

{

public tbForm()

{

InitializeComponent();

}

DataSet ds;

DB db;

NpgsqlDataAdapter adapter;

private void tbForm\_Load(object sender, EventArgs e)

{

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

db = new DB();

ds = new DataSet();

adapter = new NpgsqlDataAdapter();

db.openConnection();

NpgsqlCommand command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT taked\_books.id\_exbook, taked\_books.id\_readticket, taked\_books.val\_per\_exbook, accounts.full\_name, books.title\_book, books.author FROM taked\_books INNER JOIN accounts ON accounts.id\_readerticket = taked\_books.id\_readticket JOIN exbooks ON exbooks.id\_exbook = taked\_books.id\_exbook JOIN books ON books.id\_book = exbooks.id\_book";

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[4].Width = 200;

dataGridView1.Columns[3].Width = 200;

}

private void button2\_Click(object sender, EventArgs e)

{

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

db = new DB();

ds = new DataSet();

adapter = new NpgsqlDataAdapter();

db.openConnection();

NpgsqlCommand command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT taked\_books.id\_exbook, taked\_books.id\_readticket, taked\_books.val\_per\_exbook, accounts.full\_name, accounts.phone\_acc, books.title\_book, books.author FROM taked\_books INNER JOIN accounts ON accounts.id\_readerticket = taked\_books.id\_readticket JOIN exbooks ON exbooks.id\_exbook = taked\_books.id\_exbook JOIN books ON books.id\_book = exbooks.id\_book WHERE taked\_books.id\_readticket = @id\_readticket";

command.Parameters.Add("@id\_readticket", NpgsqlDbType.Integer).Value = int.Parse(textBox1.Text);

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[3].Width = 200;

dataGridView1.Columns[4].Width = 200;

}

private void button1\_Click(object sender, EventArgs e)

{

int id\_exbook = int.Parse(textBox2.Text);

NpgsqlCommand command = new NpgsqlCommand("DELETE FROM taked\_books WHERE id\_exbook = @id\_exbook", db.getConnection());

DateTime now = DateTime.Now.AddDays(14);

command.Parameters.Add("@id\_exbook", NpgsqlDbType.Integer).Value = id\_exbook;

db.openConnection();

command = new NpgsqlCommand("UPDATE exbooks SET id\_readticket=NULL WHERE id\_exbook=@id\_exbook", db.getConnection());

command.Parameters.Add("@id\_exbook", NpgsqlDbType.Integer).Value = id\_exbook;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Книга вернута!");

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

db = new DB();

ds = new DataSet();

adapter = new NpgsqlDataAdapter();

db.openConnection();

command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT taked\_books.id\_exbook, taked\_books.id\_readticket, taked\_books.val\_per\_exbook, accounts.full\_name, accounts.phone\_acc, books.title\_book, books.author FROM taked\_books INNER JOIN accounts ON accounts.id\_readerticket = taked\_books.id\_readticket JOIN exbooks ON exbooks.id\_exbook = taked\_books.id\_exbook JOIN books ON books.id\_book = exbooks.id\_book WHERE taked\_books.id\_readticket = @id\_readticket";

command.Parameters.Add("@id\_readticket", NpgsqlDbType.Integer).Value = int.Parse(textBox1.Text);

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[3].Width = 200;

dataGridView1.Columns[4].Width = 200;

}

}

private void button3\_Click(object sender, EventArgs e)

{

this.Hide();

AdminForm mainForm = new AdminForm();

mainForm.Show();

}

private void keypress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

}

}

}

}

Userbooks.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Npgsql;

using NpgsqlTypes;

namespace OnlineLibrary

{

public partial class Userbooks : Form

{

public Userbooks()

{

InitializeComponent();

}

public int redtic;

public string acc\_login;

private void Userbooks\_Load(object sender, EventArgs e)

{

comboBox1.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;

comboBox2.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;

comboBox3.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;

comboBox1.SelectedIndex = 0;

DB db = new DB();

DataTable dt = new DataTable();

NpgsqlCommand command = new NpgsqlCommand("SELECT books.title\_book FROM books", db.getConnection());

NpgsqlDataAdapter adapter = new NpgsqlDataAdapter("", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(dt);

if (dt.Rows.Count > 0)

{

for (int i = 0; i < dt.Rows.Count; i++)

{

DataRow row = dt.Rows[i];

string titles = row["title\_book"].ToString();

comboBox2.Items.Add(titles);

comboBox2.SelectedIndex = 0;

}

}

}

private void button1\_Click(object sender, EventArgs e)

{

int ind = comboBox1.SelectedIndex + 1;

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

DB db = new DB();

DataSet ds = new DataSet();

NpgsqlDataAdapter adapter = new NpgsqlDataAdapter();

db.openConnection();

NpgsqlCommand command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT books.title\_book, books.author FROM books INNER JOIN exbooks ON books.id\_book = exbooks.id\_book WHERE exbooks.id\_readticket IS NULL AND exbooks.id\_library = @id\_library GROUP BY books.title\_book, books.author";

command.Parameters.Add("@id\_library", NpgsqlDbType.Integer).Value = ind;

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[0].Width = 200;

dataGridView1.Columns[1].Width = 150;

}

private void button2\_Click(object sender, EventArgs e)

{

string ind = textBox1.Text;

dataGridView2.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

DB db = new DB();

DataSet ds = new DataSet();

NpgsqlDataAdapter adapter = new NpgsqlDataAdapter();

db.openConnection();

NpgsqlCommand command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT books.title\_book, books.author, exbooks.id\_library FROM books INNER JOIN exbooks ON books.id\_book = exbooks.id\_book WHERE exbooks.id\_readticket IS NULL AND title\_book LIKE @word GROUP BY books.title\_book, books.author, exbooks.id\_library";

command.Parameters.AddWithValue("@word", "%" + ind + "%");

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView2.DataSource = ds.Tables[0];

dataGridView2.Columns[0].Width = 200;

dataGridView2.Columns[1].Width = 150;

}

private void button4\_Click(object sender, EventArgs e)

{

string book\_title = comboBox2.SelectedItem.ToString();

int id\_book = 0;

DB db = new DB();

DataTable dt = new DataTable();

NpgsqlCommand command = new NpgsqlCommand("SELECT books.id\_book FROM books WHERE title\_book = (@title\_book)", db.getConnection());

command.Parameters.Add("@title\_book", NpgsqlDbType.Varchar).Value = book\_title;

NpgsqlDataAdapter adapter = new NpgsqlDataAdapter("", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(dt);

if (dt.Rows.Count > 0)

{

DataRow row = dt.Rows[0];

string titles = row["id\_book"].ToString();

id\_book = int.Parse(titles);

}

int id\_exbook = 0;

db = new DB();

dt = new DataTable();

int lib = comboBox3.SelectedIndex + 1;

command = new NpgsqlCommand("SELECT exbooks.id\_exbook FROM exbooks WHERE id\_book = @id\_book AND id\_library = @id\_library AND id\_readticket IS NULL", db.getConnection());

command.Parameters.Add("@id\_book", NpgsqlDbType.Integer).Value = id\_book;

command.Parameters.Add("@id\_library", NpgsqlDbType.Integer).Value = lib;

adapter = new NpgsqlDataAdapter("", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(dt);

if (dt.Rows.Count > 0)

{

DataRow row = dt.Rows[0];

string titles = row["id\_exbook"].ToString();

id\_exbook = int.Parse(titles);

}

db = new DB();

DataTable table = new DataTable();

adapter = new NpgsqlDataAdapter();

command = new NpgsqlCommand("UPDATE exbooks SET id\_readticket = @id\_readticket WHERE id\_exbook = @id\_exbook", db.getConnection());

command.Parameters.Add("@id\_readticket", NpgsqlDbType.Integer).Value = redtic;

command.Parameters.Add("@id\_exbook", NpgsqlDbType.Integer).Value = id\_exbook;

adapter.SelectCommand = command;

adapter.Fill(table);

try

{

db.openConnection();

NpgsqlDataAdapter da = new NpgsqlDataAdapter("", db.getConnection());

da.InsertCommand = new NpgsqlCommand("INSERT INTO not\_taked\_books (\"id\_exbook\", \"id\_readticket\", \"booking\_date\") values(:id\_exbook, :id\_readticket, :booking\_date)", db.getConnection());

da.InsertCommand.Parameters.Add(new NpgsqlParameter("id\_exbook", NpgsqlDbType.Integer));

da.InsertCommand.Parameters.Add(new NpgsqlParameter("id\_readticket", NpgsqlDbType.Integer));

da.InsertCommand.Parameters.Add(new NpgsqlParameter("booking\_date", NpgsqlDbType.Varchar));

da.InsertCommand.Parameters[0].Direction = ParameterDirection.Input;

da.InsertCommand.Parameters[1].Direction = ParameterDirection.Input;

da.InsertCommand.Parameters[2].Direction = ParameterDirection.Input;

da.InsertCommand.Parameters[0].SourceColumn = "id\_exbook";

da.InsertCommand.Parameters[1].SourceColumn = "id\_readticket";

da.InsertCommand.Parameters[2].SourceColumn = "booking\_date";

da.InsertCommand.Parameters[0].Value = id\_exbook;

da.InsertCommand.Parameters[1].Value = redtic;

da.InsertCommand.Parameters[2].Value = DateTime.Now.AddDays(10).ToString();

da.InsertCommand.ExecuteNonQuery();

MessageBox.Show("Книга выбрана! Она будет ждать вас в библиотеке в течение 10 дней!");

}

catch

{

MessageBox.Show("Этой книги нет в выбранной библиотеке!");

}

}

private void button3\_Click(object sender, EventArgs e)

{

this.Hide();

UserForm mainForm = new UserForm();

mainForm.acc\_login = acc\_login;

mainForm.Show();

}

}

}

**userForm.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Npgsql;

using NpgsqlTypes;

namespace OnlineLibrary

{

public partial class UserForm : Form

{

public UserForm()

{

InitializeComponent();

}

public string acc\_login;

NpgsqlDataAdapter adapter;

int redtic = 0;

string per = "";

private void UserForm\_Load(object sender, EventArgs e)

{

DB db = new DB();

DataTable dt = new DataTable();

dt = new DataTable();

NpgsqlCommand command = new NpgsqlCommand("SELECT accounts.full\_name FROM accounts WHERE login = (@login)", db.getConnection());

command.Parameters.Add("@login", NpgsqlDbType.Varchar).Value = acc\_login;

adapter = new NpgsqlDataAdapter("", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(dt);

if (dt.Rows.Count > 0)

{

DataRow row = dt.Rows[0];

string titles = row["full\_name"].ToString();

label1.Text = titles;

}

db = new DB();

dt = new DataTable();

dt = new DataTable();

command = new NpgsqlCommand("SELECT accounts.id\_readerticket FROM accounts WHERE login = (@login)", db.getConnection());

command.Parameters.Add("@login", NpgsqlDbType.Varchar).Value = acc\_login;

adapter = new NpgsqlDataAdapter("", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(dt);

if (dt.Rows.Count > 0)

{

DataRow row = dt.Rows[0];

string titles = row["id\_readerticket"].ToString();

label2.Text = "Номер читательского билета: " + titles;

redtic = int.Parse(titles);

}

db = new DB();

dt = new DataTable();

dt = new DataTable();

command = new NpgsqlCommand("SELECT reader\_tickets.val\_per\_rt FROM reader\_tickets WHERE id\_readerticket = (@id\_readerticket)", db.getConnection());

command.Parameters.Add("@login", NpgsqlDbType.Integer).Value = redtic;

adapter = new NpgsqlDataAdapter("", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(dt);

if (dt.Rows.Count > 0)

{

DataRow row = dt.Rows[0];

string titles = row["val\_per\_rt"].ToString();

label6.Text = "Срок действия: " + titles;

per = titles;

}

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

db = new DB();

DataSet ds = new DataSet();

adapter = new NpgsqlDataAdapter();

db.openConnection();

command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT exbooks.id\_library, books.title\_book, books.author, taked\_books.val\_per\_exbook FROM books INNER JOIN exbooks ON books.id\_book = exbooks.id\_book JOIN taked\_books ON exbooks.id\_exbook = taked\_books.id\_exbook JOIN accounts ON exbooks.id\_readticket = accounts.id\_readerticket WHERE accounts.id\_readerticket = @id\_readerticket";

command.Parameters.Add("@id\_readerticket", NpgsqlDbType.Integer).Value = redtic;

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[0].Width = 200;

dataGridView1.Columns[1].Width = 150;

dataGridView2.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

db = new DB();

ds = new DataSet();

adapter = new NpgsqlDataAdapter();

db.openConnection();

command = new NpgsqlCommand();

command.Connection = db.getConnection();

command.CommandType = CommandType.Text;

command.CommandText = "SELECT exbooks.id\_library, books.title\_book, books.author, not\_taked\_books.booking\_date FROM books INNER JOIN exbooks ON books.id\_book = exbooks.id\_book JOIN not\_taked\_books ON exbooks.id\_exbook = not\_taked\_books.id\_exbook JOIN accounts ON exbooks.id\_readticket = accounts.id\_readerticket WHERE accounts.id\_readerticket = @id\_readerticket";

command.Parameters.Add("@id\_readerticket", NpgsqlDbType.Integer).Value = redtic;

adapter.SelectCommand = command;

adapter.Fill(ds);

dataGridView2.DataSource = ds.Tables[0];

dataGridView2.Columns[0].Width = 200;

dataGridView2.Columns[1].Width = 150;

textBox1.ReadOnly = true;

}

private void button1\_Click(object sender, EventArgs e)

{

this.Hide();

Autorisation mainForm = new Autorisation();

mainForm.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

DateTime myDate = DateTime.ParseExact(per, "dd-MM-yyyy HH:mm:ss", System.Globalization.CultureInfo.InvariantCulture);

if (myDate < DateTime.Now)

{

MessageBox.Show("Ваш читательский билет недействителен!");

}

else

{

this.Hide();

Userbooks mainForm = new Userbooks();

mainForm.redtic = redtic;

mainForm.acc\_login = acc\_login;

mainForm.Show();

}

}

}

}