Министерство образования и науки Российской Федерации

**Муромский институт (филиал)**

федерального государственного бюджетного образовательного учреждения высшего образования

**«Владимирский государственный университет**

**Имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(МИ ВлГУ)**

Факультет ИТР

Кафедра: ПИн

Курсовая работа

По: *Системы управления базами данных*

Тема: АИС система библиотеки

Руководитель



Колпаков А.А.

(Оценка)

(фамилия, инициалы)

(подпись) (дата)

Члены комиссии

Студент ПИН-122

(группа)

Чарушина Е.В.

(Подпись Ф.И.О.)

(фамилия, инициалы)

(Подпись Ф.И.О.)

(подпись) (дата)

Муром 2024

Муромский институт (филиал) федерального государственного бюджетного образовательного учреждения высшего образования

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет Информационных Технологий и Радиоэлектроники

«УТВЕРЖДАЮ»

Зав. кафедрой ПИн \_\_\_\_\_\_\_\_\_\_\_А.Л. Жизняков

(подпись)

*«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.*

**ЗАДАНИЕ**

**На курсовую работу по курсу  *Системы управления базами данных .***

Студент Чарушина Е.В. гр. ПИН-122 .

1. Тема работы АИС система библиотеки

|  |
| --- |
|  |

2. Сроки сдачи студентом законченного проекта  *« » декабря 2024г.*

3. Исходные данные к работе  *...*

- В БД содержатся сведения об абонентах библиотеки, имеющейся литературе и ее хранении. Необходимо вести учет выданной литературы. В каталоге на каждое название книги выписывается каталожная карточка со всеми реквизитами. В БД могут быть таблицы по пунктам каталожной карточки, причем обязательно должна быть информация о количестве экземпляров каждой книги. .

На каждый экземпляр книги выписывается формуляр с основными реквизитами, записями о выдаче читателю и возврате. .

На каждого читателя заполняется формуляр с перечнем полученных книг и ставится дата их возврата. В БД должна быть информация о всех читателях со всеми данными абонементной карточки. .

- В БД предусмотреть хранение изображений (минимум в одном поле) в соответствии с тематикой курсовой работы. .

- В БД должны быть реализованы хранимые процедуры и/или триггеры. .

- в БД должно быть занесено суммарно не менее 50 записей .

**Содержание**

[**Введение** 4](#_Toc186159678)

[**1. Анализ технического задания** 5](#_Toc186159679)

[**2. Разработка моделей данных** 8](#_Toc186159680)

[**3. Разработка и реализация АИС** 12](#_Toc186159681)

[**3.1 Структура базы данных** 12](#_Toc186159682)

[**3.2 Взаимодействие с базой данных** 13](#_Toc186159683)

[**3.3 Структура программы** 13](#_Toc186159684)

[**3.4 Инструкция пользователя** 33](#_Toc186159685)

[**4. Тестирование АИС** 38](#_Toc186159686)

[**4.1 Тестирование базы данных** 38](#_Toc186159687)

[**4.2 Тестирование программы** 47](#_Toc186159688)

[**Заключение** 51](#_Toc186159689)

[**Список литературы** 52](#_Toc186159690)

[**Приложения** 53](#_Toc186159691)

[**Приложение А** 53](#_Toc186159692)

[**Приложение Б** 54](#_Toc186159693)

[**Приложение В** 55](#_Toc186159694)

**Введение**

Вступление: В современном мире библиотеки должны адаптироваться к быстро меняющимся условиям и ожиданиям пользователей. Внедрение автоматизированных информационных систем (АИС) становится важным для обеспечения удобного доступа к ресурсам. АИС оптимизируют процессы учета, что повышает качество библиотечных услуг и улучшает организационную структуру библиотек.

Актуальность работы: Современные библиотеки нуждаются в оптимизации учета, хранения и выдачи книг, а также в улучшении взаимодействия с читателями. Автоматизация этих процессов существенно повышает эффективность работы библиотек.

Цель работы: Разработать АИС для библиотеки

Задачи:

* Проанализировать существующие АИС в библиотеке и выявить их преимущества и недостатки;
* Определить функциональные требования к разрабатываемой системе;
* Разработать архитектуру автоматизированной информационной системы для библиотеки;
* Описать основные алгоритмы работы системы.

Объект исследования: АИС библиотеки

Предмет исследования: Проектирование АИС библиотеки

Методы, используемые в работе:

* Анализ литературы по теме автоматизации библиотечных процессов;
* Сравнительный анализ существующих АИС;
* Метод проектирования для разработки архитектуры системы;
* Метод моделирования для описания алгоритмов работы системы.

Теоретическая важность: Заключается в систематизации знаний о современных подходах к автоматизации библиотечных процессов и разработке АИС

Практическая важность: Заключается в применении разработанной АИС в библиотеках, что улучшит качество обслуживания пользователей, оптимизирует внутренние процессы и повысит общую эффективность работы библиотек.

Структура работы: В первой части работы представлен анализ требований к системе и её описание. Во второй части работы содержатся модели АИС и примеры её работы.

**1. Анализ технического задания**

Согласно техническому заданию на курсовую работу необходимо разработать АИС библиотеки.

АИС библиотеки предназначена для автоматизации процессов управления библиотечным фондом, учета читателей, выдачи и возврата книг. Основными пользователями системы являются библиотекари и читатели. Система должна обеспечивать эффективное взаимодействие между этими группами.

Для успешного проектирования АИС библиотеки необходимо провести исследование потребностей потенциальных пользователей. Библиотекари нуждаются в удобном интерфейсе для учета книг, автоматизации процессов выдачи и возврата. Читатели хотят иметь возможность быстро находить и бронировать книги, а также создавать запрос на увеличение книжного фонда библиотеки.

Ожидается снижение трудозатрат на выполнение рутинных операций, повышение качества обслуживания читателей и оптимизация управления библиотечным фондом.

Требования к базе данных:

* должны содержаться сведения о читателях, авторах, имеющейся литературе, выдаче и возврате книг, а также о заказываемой литературе в библиотеку;
* должно быть предусмотрено хранение изображений;
* должны быть реализованы хранимые процедуры или триггеры;
* должно быть не менее 50 записей.

Требования к приложению:

* должно обеспечивать надежное и безопасное соединение с базой данных;
* должно иметь интуитивно понятный и удобный графический интерфейс;
* должно включать механизмы для автоматической диагностики ошибок, предоставляя пользователю понятные сообщения об ошибках;
* должны быть реализованы классы и методы, позволяющие эффективно извлекать, изменять и удалять информацию из таблиц базы данных;
* должно вести временной учёт забронированной литературы.

Для данной работы была выбрана база данных PostgreSQL[2]. Рассмотрим основные критерии выбора:

* поддерживает реляционную модель данных, что обеспечит адекватное представление информации о книгах, читателях и операциях;
* обеспечивает высокую производительность обработки запросов даже при увеличении объема данных (например, при добавлении новых книг или читателей);
* позволяет легко масштабировать систему и добавлять новые функции в будущем (например, интеграция с электронными ресурсами);
* является бесплатной и открытой СУБД, что снижает затраты на лицензирование.

Для более обоснованного выбора рассмотрим также две альтернативные базы данных: MySQL и Microsoft SQL Server. MySQL:

* реляционная модель данных: MySQL также поддерживает реляционную модель данных, однако в некоторых случаях может иметь ограничения по функциональности по сравнению с PostgreSQL, особенно в области поддержки сложных запросов и транзакций;
* производительность: MySQL известен своей высокой производительностью при чтении данных, что делает его отличным выбором для приложений, ориентированных на операции чтения. Однако при сложных запросах и больших объемах данных его производительность может уступать PostgreSQL;
* лицензирование: MySQL также имеет бесплатную версию (Community Edition), но для доступа к полному набору функций может потребоваться платная версия, что увеличивает затраты на лицензирование;
* реляционная модель данных: Microsoft SQL Server поддерживает реляционную модель данных и предлагает широкий спектр функций для работы с данными, включая продвинутые аналитические возможности;
* SQL Server обеспечивает высокую производительность и оптимизацию запросов, но в некоторых случаях может требовать значительных ресурсов сервера, особенно при больших объемах данных;
* Microsoft SQL Server является коммерческой СУБД, что подразумевает значительные затраты на лицензирование.
* В заключение, выбор PostgreSQL обоснован его высокой производительностью, масштабируемостью, поддержкой реляционной модели данных и отсутствием затрат на лицензирование. Эти факторы делают PostgreSQL идеальным выбором для разработки системы управления библиотекой.

При выборе технологий для создания приложения важно учитывать их особенности и преимущества по сравнению с альтернативами.

Windows Forms против WPF. Хотя WPF (Windows Presentation Foundation) является более современным решением для создания настольных приложений с поддержкой MVVM и богатых графических интерфейсов, Windows Forms[3] предоставляет более простую и интуитивно понятную среду разработки. Для небольших или средних приложений, которые требуют быстрого прототипирования и разработки, Windows Forms может оказаться более подходящим выбором. Кроме того, наличие большого количества готовых компонентов и библиотек для Windows Forms позволяет ускорить процесс разработки.

Npgsql против Entity Framework. Npgsql, как .NET Data Provider для PostgreSQL[1], предоставляет более низкоуровневый доступ к базе данных, чем Entity Framework (EF). Хотя EF облегчает работу с базами данных, используя ORM (Object-Relational Mapping), он может добавлять лишнюю абстракцию и снижение производительности в некоторых случаях. Если ваше приложение требует быстрого доступа к данным и высокой производительности, Npgsql позволяет тонко настраивать запросы и управлять транзакциями более эффективно.

Quartz против Hangfire. Quartz и Hangfire — это обе мощные библиотеки для планирования задач. Но Quartz предлагает более гибкие возможности для настройки расписания и управления задачами в распределенных системах. Hangfire, с другой стороны, прост в использовании и хорошо интегрируется с ASP.NET. Однако, в случае настольного приложения, Quartz предоставляет более богатый функционал для планирования повторяющихся задач и их управления.

Выбор C# Windows Forms, Npgsql и Quartz для разработки приложения обусловлен их удобством, производительностью и необходимыми функциональными возможностями. Эти технологии обеспечивают эффективное сочетание простоты и мощности, что особенно важно для учета должников книг.

**2. Разработка моделей данных**

В данной работе инфологическое проектирование построено на функциональном подходе к проектированию БД. Этот метод реализует принцип "от задач" и применяется тогда, когда известны функции некоторой группы лиц и/или комплекса задач, для обслуживания информационных потребностей которых создаётся рассматриваемая БД.

Функциональные требований к системе описаны в формате UseCase (рисунок 1).

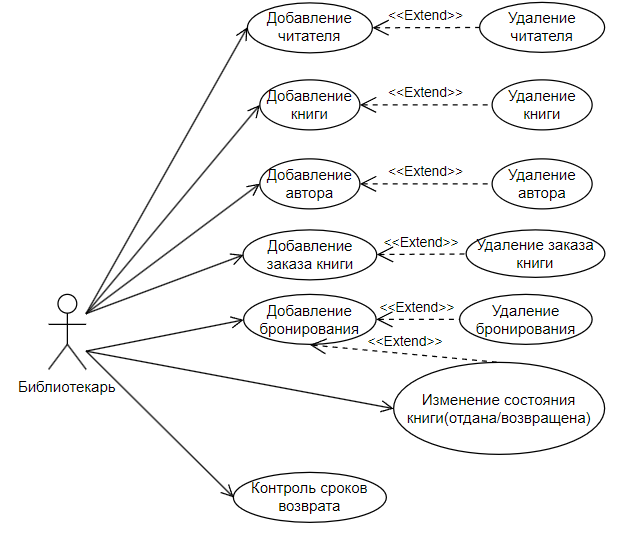


Рисунок 1 – UseCase диаграмма для АИС библиотеки.

На этапе логического проектирования была создана логическая структура базы данных, которая представлена на рисунке А.1. Количество полей таблиц и их наименование составлялось в соответствии с библиотечным гостом[5].

На основе логической модели базы данных была создана физическая модель (рисунок Б.1). Поскольку в логической модели присутствует связь многие ко многим между таблицами «Каталог авторов» и «Каталог книг» в базе данных была создана промежуточная таблица, которая содержит идентификаторы книг и их авторов.

Доступ к базе данных имеет только библиотекарь путём взаимодействия через приложение. Соответственно защита данных пользователя обеспечивается законом о персональных данных[4].

Далее представлены таблицы, описывающие структуру таблиц базы данных.

Таблица 1. Пояснение к таблице Authors\_catalog

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| id | Идентификатор записи. Первичный ключ таблицы |
| author\_surname | Фамилия автора |
| author\_name | Имя автора |
| author\_patronymic | Отчество автора (не обязательно для заполнения) |

Таблица 2. Пояснение к таблице Books\_catalog

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| index | Индекс книги. Первичный ключ таблицы |
| authors\_mark | Авторская метка |
| title | Название книги |
| place\_publication | Место публикации (город) |
| information\_publication | Информация об издании (название) |
| volume | Объём книги (количество страниц) |
| quantity\_total | Общее количество книг данного экземпляра (полученные библиотекой) |
| quantity\_remaining | Количество книг, которые сейчас в библиотеке |
| cover | Фотография обложки (не обязательно для заполнения) |
| date\_publication | Дата публикации |

Таблица 3. Пояснение к таблице Readers\_catalog

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| id | Номер читательского билета. Первичный ключ таблицы |
| surname | Фамилия читателя |
| name | Имя читателя |
| patronymic | Отчество читателя (не обязательно для заполнения) |
| birthday | Дата рождения |
| education | Образование (среднее, высшее и т.д. не обязательно для заполнения) |
| profession | Профессия (не обязательно для заполнения) |
| educational\_inst | Место учёбы (школа, институт и т.д. не обязательно для заполнения) |
| city | Город проживания |
| street | Улица проживания |
| house | Номер дома проживания |
| building\_house | Корпус дома проживания (не обязательно для заполнения) |
| flat | Квартира проживания (не обязательно для заполнения) |
| passport\_series | Серия паспорта |
| passport\_number | Номер паспорта |
| issued\_by\_whom | Кем выдан паспорт |
| date\_issue | Дата выдачи паспорта |
| consists\_of | Дата, с которой состоит читателем |
| re\_registration | Дата перерегистрации (не обязательно для заполнения) |
| phone | Номер телефона читателя |

Таблица 4. Пояснение к таблице Authors\_Books

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| id | Идентификатор записи. Первичный ключ таблицы |
| author\_id | Идентификатор автора. Внешний ключ |
| book\_id | Индекс книги. Внешний ключ |

Таблица 5. Пояснение к таблице Booking\_catalog

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| index | Индекс книги. Внешний ключ |
| reader\_id | Идентификатор читателя. Внешний ключ |
| quantity | Количество книг |
| date\_issue | Запланированная дата выдачи книги |
| date\_return | Запланированная дата возврата книги |
| id | Номер бронирования. Первичный ключ |
| issued | Поле логического типа. Значение true – книга выдана, false – не выдана |
| returned | Поле логического типа. Значение true – книга возвращена, false – не возвращена |

Таблица 6. Пояснение к таблице Order\_catalog

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| id | Номер заказа. Первичный ключ |
| title | Название книги |
| author\_surname | Фамилия автора |
| author\_name | Имя автора (не обязательно для заполнения) |
| author\_patronymic | Отчество автора (не обязательно для заполнения) |
| quantity | Количество книг |
| reader\_id | Идентификатор читателя |
| date\_publication | Дата публикации (не обязательно для заполнения) |

**3. Разработка и реализация АИС**

**3.1 Структура базы данных**

В базе данных всего 6 таблиц, описание которых приводилось в предыдущем пункте. Также в базе данных были созданы триггер trigger\_deduct\_quantity и триггерная функция deduct\_quantity\_remaining. Триггер срабатывает после добавления записи в таблицу Booking\_catalog или после её редактирования. Он запускает функцию, которая проверяет поля issued и returned таблицы Booking\_catalog на изменение значения false на true. Если условие выполнилось для поля issued, происходит уменьшение значения поля quantity\_remaining таблицы Books\_catalog на значение quantity таблицы Booking\_catalog. Если условие выполнилось для поля returned, происходит увеличение значения поля quantity\_remaining таблицы Books\_catalog на значение quantity таблицы Booking\_catalog. Это необходимо для того, чтобы автоматизировать процесс учёта количества книг, находящихся в библиотеке.

Код создания триггера trigger\_deduct\_quantity

CREATE OR REPLACE TRIGGER trigger\_deduct\_quantity

AFTER INSERT OR UPDATE

ON kursovaya."Booking\_catalog"

FOR EACH ROW

EXECUTE FUNCTION public.deduct\_quantity\_remaining();

Код создания триггерной функции deduct\_quantity\_remaining

CREATE OR REPLACE FUNCTION deduct\_quantity\_remaining()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

AS $BODY$

BEGIN

-- Если книга выдана, уменьшаем количество оставшихся

IF TG\_OP = 'INSERT' and new.issued = true or (OLD.issued = false AND NEW.issued =true) then

UPDATE kursovaya."Books\_catalog"

SET quantity\_remaining = quantity\_remaining - NEW.quantity

WHERE index = NEW.index;

-- Если книга возвращена, увеличиваем количество оставшихся

ELSIF NEW.returned THEN

UPDATE kursovaya."Books\_catalog"

SET quantity\_remaining = quantity\_remaining + new.quantity

WHERE index = new.index;

END IF;

RETURN NEW;

END;

$BODY$;

ALTER FUNCTION public.deduct\_quantity\_remaining()

OWNER TO postgres;

**3.2 Взаимодействие с базой данных**

Для взаимодействия с базой данных из программы происходит с помощью команд библиотеки Npgsql.

Соединение с базой данных происходит следующим образом. Устанавливается строка соединения:

string connectionString = "Host = localhost; Username = postgres; Password = 1234; Database = newDB"

Далее выполняется команда:

var connection = new NpgsqlConnection(connectionString)

Для взаимодействия с базой данных используются простые запросы. Для получения данных из таблицы используется SELECT запрос, для обновления таблицы – UPDATE запрос, для вставки новых значений в таблицу – INSERT запрос и для удаления записи – DELETE запрос.

**3.3 Структура программы**

В программе содержится 12 форм и 15 разработанных классов. Сначала рассмотрим структуру и предназначение форм.

Форма Menu содержит 5 кнопок и текстовое поле (рисунок 2).

* в текстовом поле содержится сообщение для пользователя;
* кнопка «Читатели» открывает форму Readers\_form;
* кнопка «Бронирование» открывает форму Bookings\_form;
* кнопка «Заказы» открывает форму Orders\_form;
* кнопка «Книги» открывает форму Books\_form;
* кнопка «Авторы» открывает форму Authors\_form.

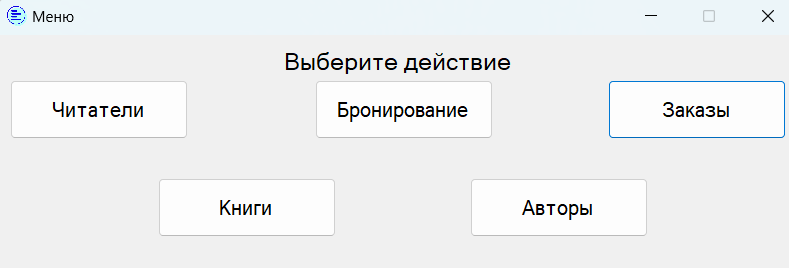


Рисунок 2 – Форма Menu

Форма Readers\_form содержит 2 текстовых поля, элемент DataGridView и 5 кнопок (рисунок 3).

* текстовые поля информируют пользователя о том, что представлено в таблице и что нужно сделать;
* элемент DataGridView выводит информацию о читателях;
* кнопки «Редактировать» и «Добавить» открывают форму Reader\_form;
* кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;
* кнопка «Бронь» открывает форму Bookings\_form;
* кнопка «Заказ» открывает форму Orders\_form.

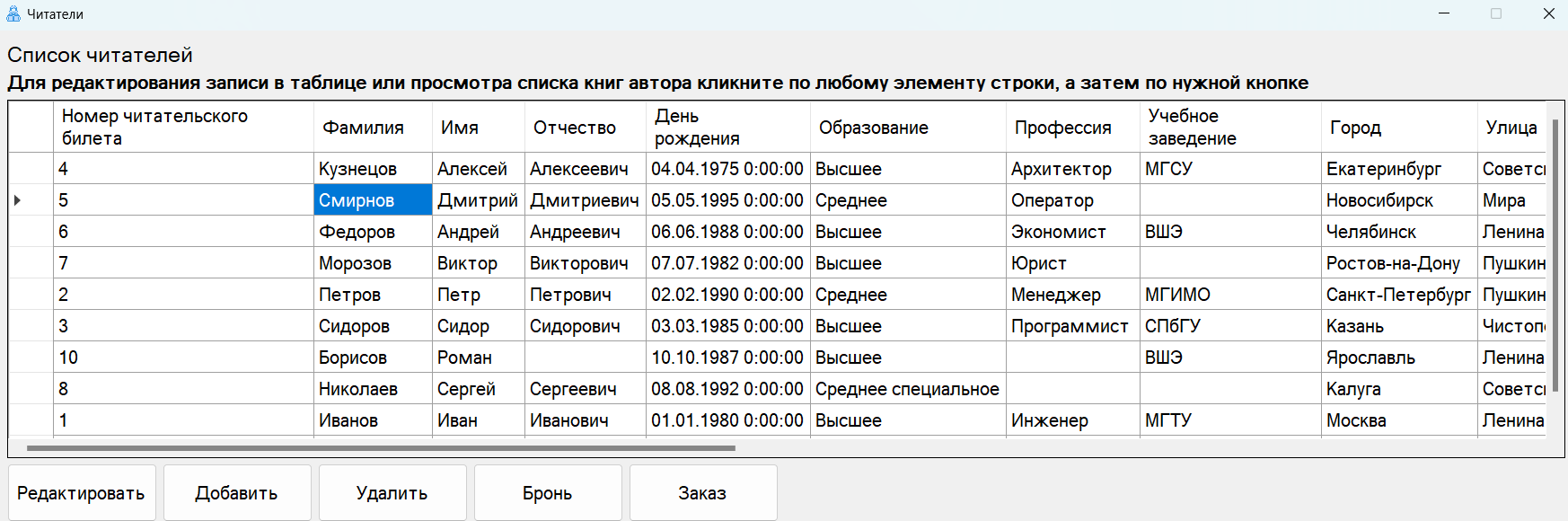


Рисунок 3 – Форма Readers\_form

Форма Reader\_form содержит 38 текстовых полей и 2 кнопки (рисунок 4).

* половина текстовых полей являются информирующими читателя, что нужно писать в другую половину текстовых полей;
* кнопка «Сохранить изменения» сохраняет информацию из текстовых полей в базе данных;
* кнопка «Пароль» генерирует логин читателя и открывает форму PasswordInput\_form.

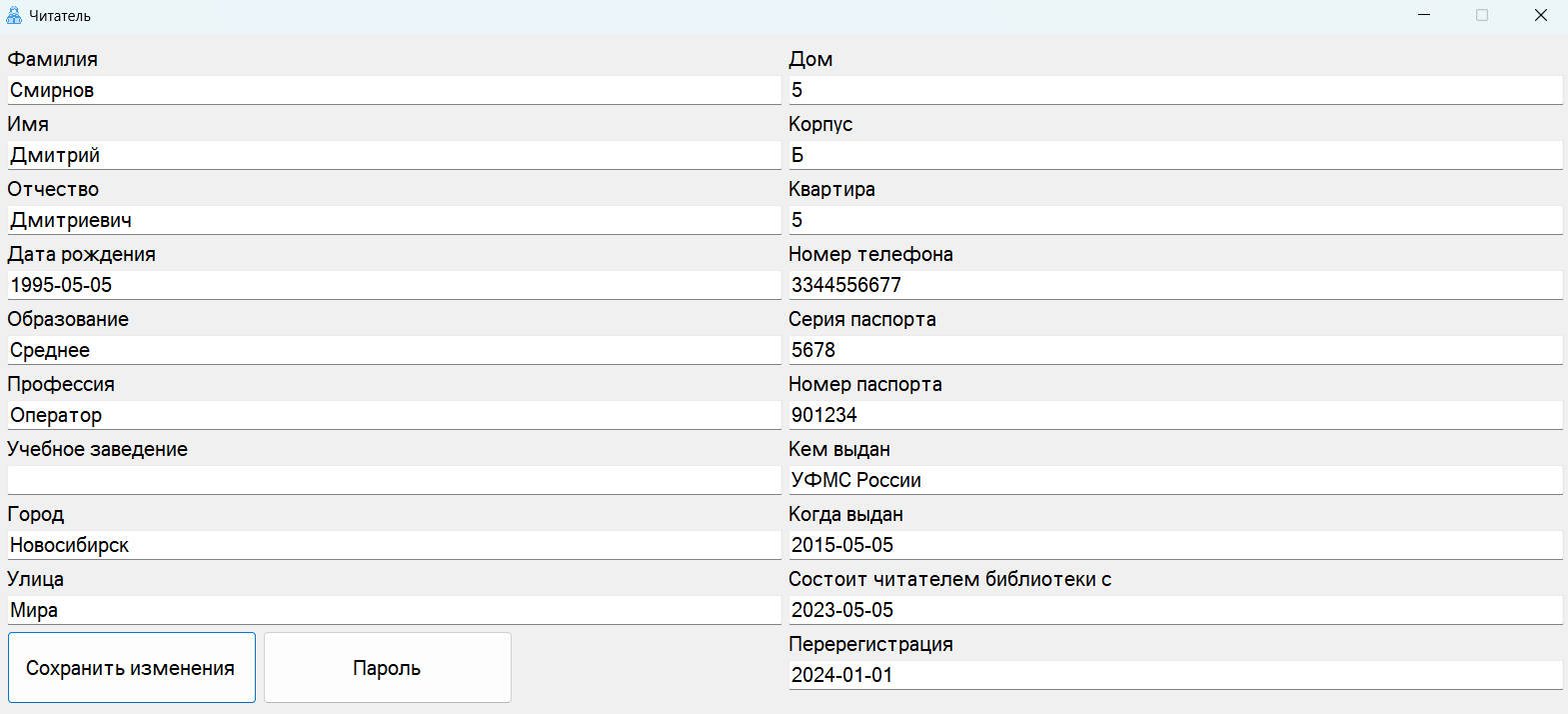


Рисунок 4 – Форма Reader\_form

Форма PasswordInput\_form содержит 2 текстовых поля и 1 кнопку (рисунок 5).

* первое текстовое поле описывает второе, в которое вводится пароль;
* кнопка «Сохранить изменения» сохраняет информацию из текстового поля в базе данных.

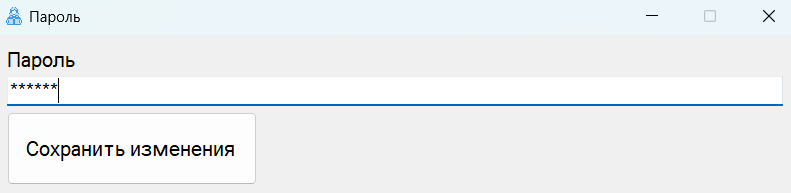


Рисунок 5 – Форма PasswordInput\_form

Форма Bookings\_form содержит 2 текстовых поля, элемент DataGridView и 4 кнопки (рисунок 6)

* текстовые поля информируют пользователя о том, что представлено в таблице и что нужно сделать;
* элемент DataGridView выводит информацию о бронировании;
* кнопки «Редактировать» и «Добавить» открывают форму Booking\_form;
* кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;
* кнопка «Читатель» открывает форму Readers\_form.

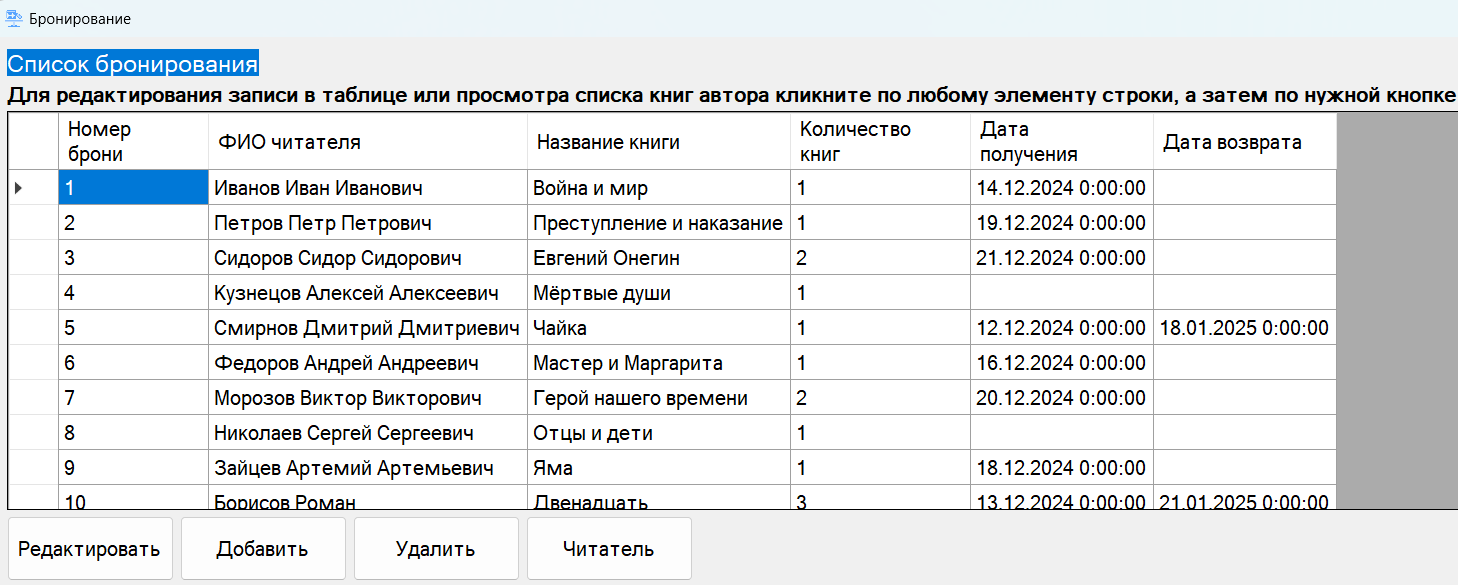


Рисунок 6 – Форма Bookings\_form

Форма Booking\_form содержит 12 текстовых полей, 2 элемента ChechBox и 3 кнопки (рисунок 7).

* 7 текстовых полей являются информационными, остальные текстовые поля используются для внесения данных;
* элементы ChechBox предназначены для изменения полей с логическим значением;
* кнопка «Сохранить изменения» сохраняет информацию из текстовых полей в базе данных;
* кнопка «Книги» открывает форму Books\_form;
* кнопка «Читатели» открывает форму Readers\_form.

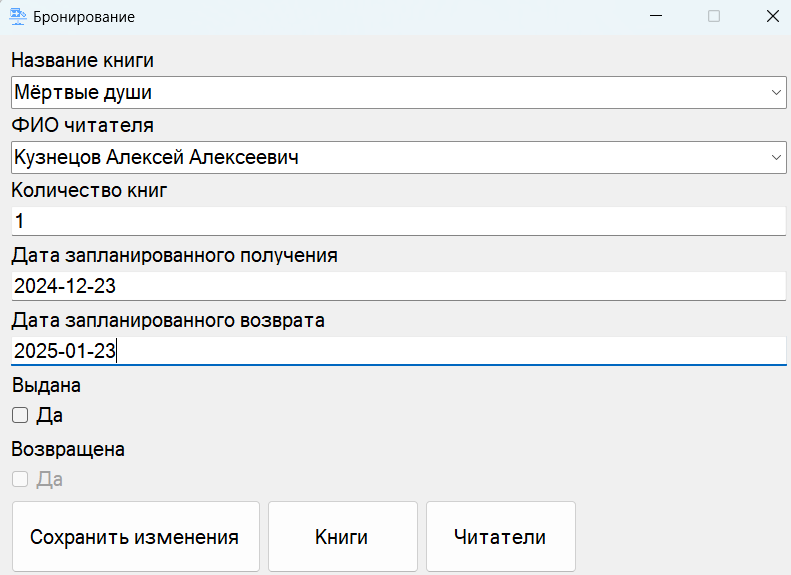


Рисунок 7 – Форма Booking\_form

Форма Books\_form содержит 2 текстовых поля, элемент DataGridView и 4 кнопки (рисунок 8).

* текстовые поля информируют пользователя о том, что представлено в таблице и что нужно сделать;
* элемент DataGridView выводит информацию о книгах;
* кнопки «Редактировать» и «Добавить» открывают форму Book\_form;
* кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;
* кнопка «Авторы» открывает форму Authors\_form.

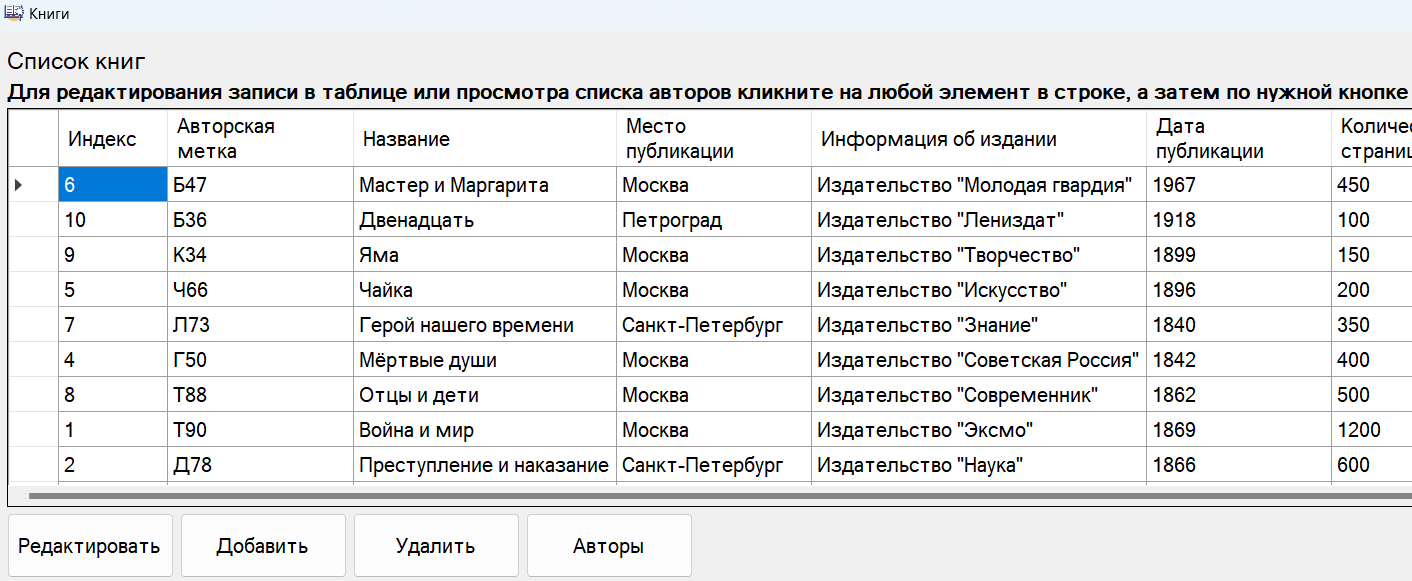


Рисунок 8 – Форма Books\_form

Форма Book\_form содержит 18 текстовых полей, 1 элемент PictureBox и 1 кнопку (рисунок 9).

* 9 текстовых полей являются информационными, остальные текстовые поля используются для внесения данных;
* элемент PictureBox выводит или изменяет картинку, хранящуюся в базе данных;
* кнопка «Сохранить изменения» сохраняет информацию из текстовых полей в базе данных.

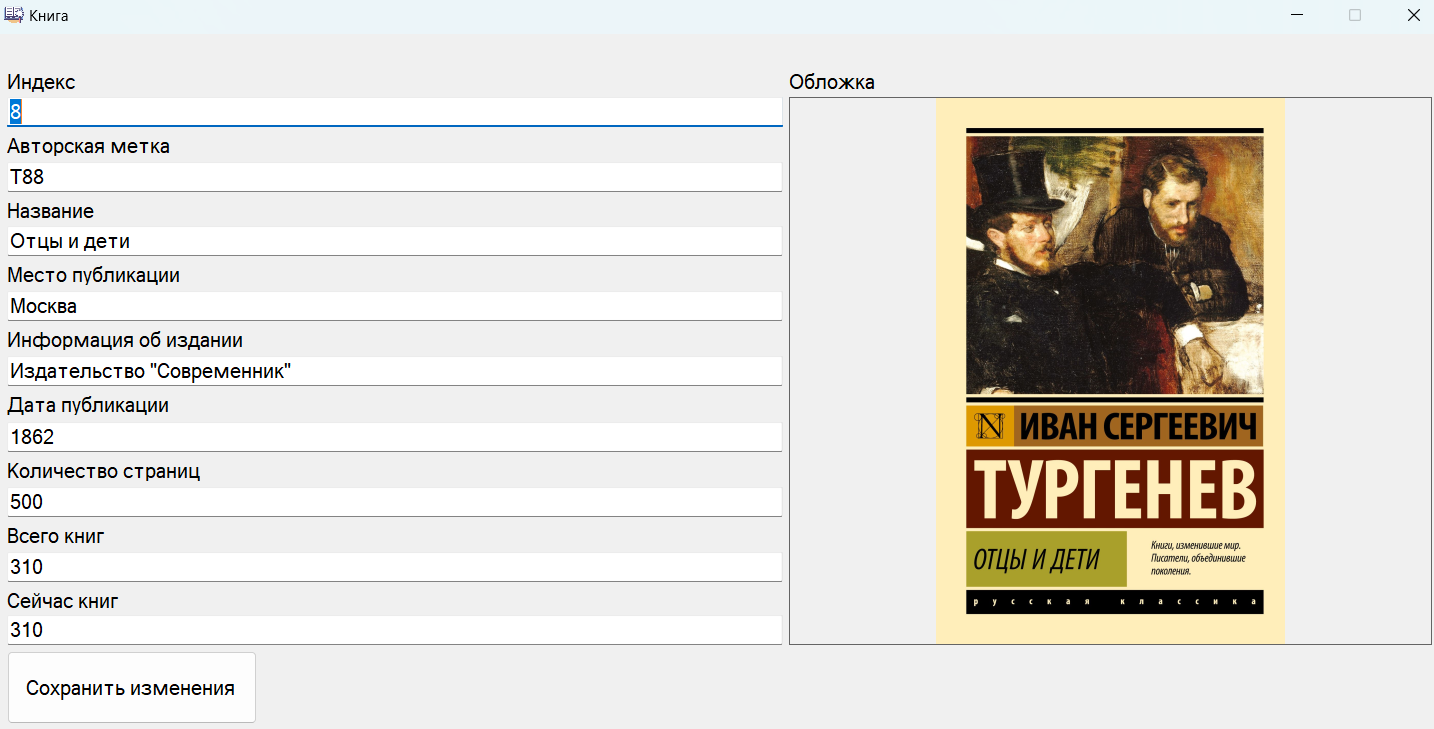


Рисунок 9 – Форма Book\_form

Форма Authors\_form содержит 2 текстовых поля, элемент DataGridView и 4 кнопки (рисунок 10).

* текстовые поля информируют пользователя о том, что представлено в таблице и что нужно сделать;
* элемент DataGridView выводит информацию об авторах;
* кнопки «Редактировать» и «Добавить» открывают форму Author\_form;
* кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;
* кнопка «Книги» открывает форму Books\_form.

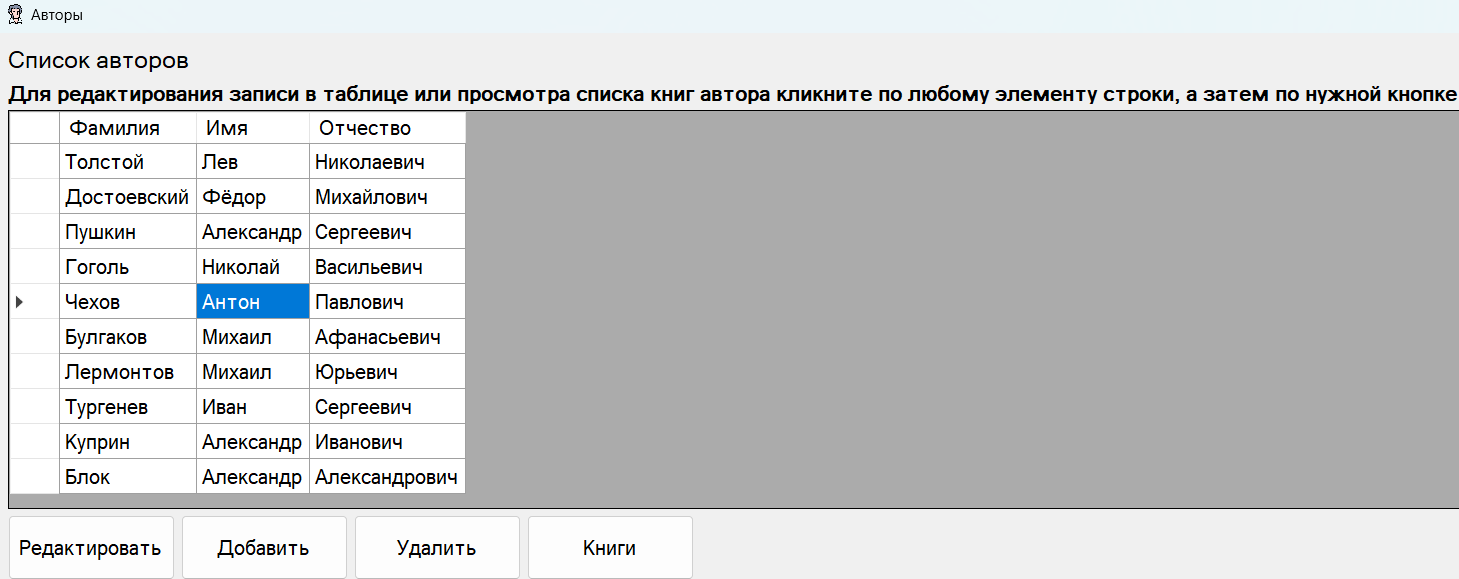


Рисунок 10 – Форма Authors\_form

Форма Author\_form содержит 6 текстовых полей и 1 кнопку (рисунок 11).

* 3 текстовых поля являются информационными, остальные текстовые поля используются для внесения данных;
* кнопка «Сохранить изменения» сохраняет информацию из текстовых полей в базе данных.

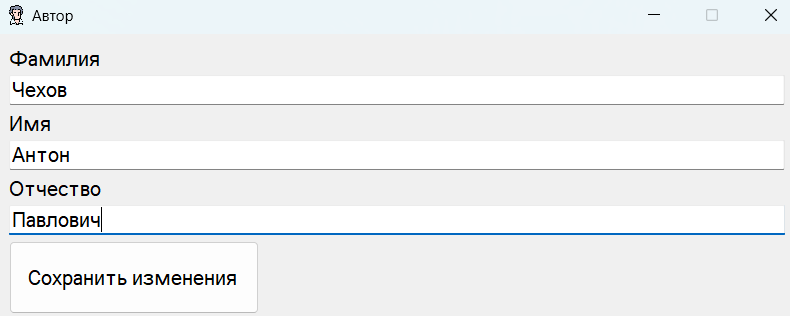


Рисунок 11 – Форма Author\_form

Форма Orders\_form содержит 2 текстовых поля, элемент DataGridView и 4 кнопки (рисунок 12).

* текстовые поля информируют пользователя о том, что представлено в таблице и что нужно сделать;
* элемент DataGridView выводит информацию о заказах;
* кнопки «Редактировать» и «Добавить» открывают форму Order\_form;
* кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;
* кнопка «Читатель» открывает форму Readers\_form.



Рисунок 12 – Форма Orders\_form

Форма Order\_form содержит 14 текстовых полей и 2 кнопки (рисунок 13).

* 7 текстовых полей являются информационными, остальные текстовые поля используются для внесения данных;
* кнопка «Сохранить изменения» сохраняет информацию из текстовых полей в базе данных;
* кнопка «Читатели» открывает форму Readers\_form.

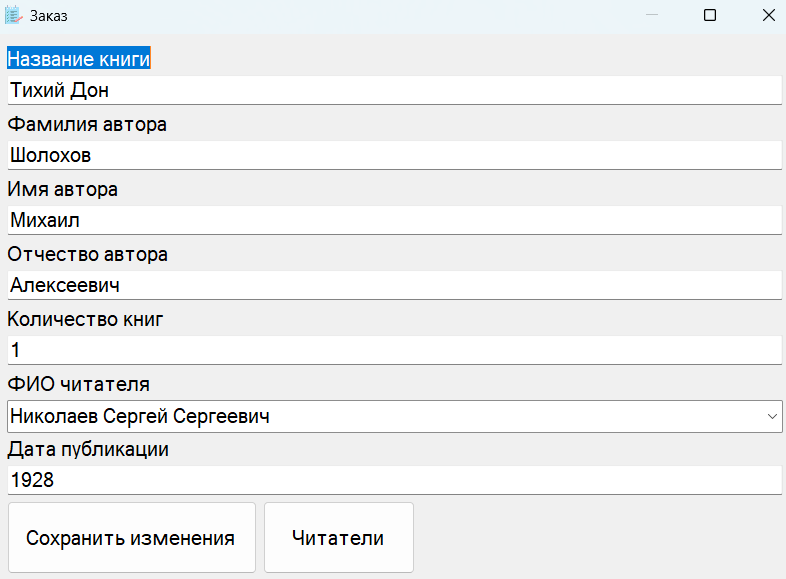


Рисунок 13 – Форма Order\_form

Класс Author

public class Author

Поля

public int id;

Хранит идентификатор записи из таблицы Authors\_catalog

public string name;

Хранит имя автора из таблицы Authors\_catalog

public string surname;

Хранит фамилию автора из таблицы Authors\_catalog

public string patronymic;

Хранит отчество автора из таблицы Authors\_catalog

Конструктор

public Author(int id, string surname, string name, string patronymic)

Параметры

int id

string surname

string name

string patronymic

Класс Booking

public class Booking

Поля

public int index\_;

Хранит индекс книги из таблицы Booking\_catalog

public int reader;

Хранит идентификатор читателя из таблицы Booking\_catalog

public int quantity\_;

Хранит количество книг из таблицы Booking\_catalog

public DateTime issue;

Хранит дату выдачи из таблицы Booking\_catalog

public DateTime return\_;

Хранит дату возврата из таблицы Booking\_catalog

public int id\_;

Хранит номер бронирования из таблицы Booking\_catalog

public bool issued;

Хранит логическое значение поля issued из таблицы Booking\_catalog

public bool returned;

Хранит логическое значение поля returned из таблицы Booking\_catalog

Конструктор

public Booking(int index\_, int reader, int quantity\_, DateTime issue, DateTime return\_, int id\_, bool issued, bool returned)

Параметры

int index\_

int reader

int quantity\_

DateTime issue

DateTime return\_

int id\_

bool issued

bool returned

Класс Book

public class Book

Поля

public int index\_;

Хранит индекс из таблицы Books\_catalog

public string mark;

Хранит авторскую марку из таблицы Books\_catalog

public string title\_;

Хранит название книги из таблицы Books\_catalog

public string place;

Хранит место публикации из таблицы Books\_catalog

public string info;

Хранит информацию о месте публикации из таблицы Books\_catalog

public int volume\_;

Хранит объём книги из таблицы Books\_catalog

public int total;

Хранит общее количество книг из таблицы Books\_catalog

public int now;

Хранит количество книг на данный момент из таблицы Books\_catalog

public byte[] cover\_;

Хранит обложку из таблицы Books\_catalog

public string date;

Хранит дату публикации из таблицы Books\_catalog

Конструктор

public Book(int index\_, string mark, string title\_, string place, string info, int volume\_, int total, int now, byte[] cover\_, string date)

Параметры

int index\_

string mark

string title\_

string place

string info

int volume\_

int total

int now

byte[] cover\_

string date

Класс Reader

public class Reader

Поля

public int id\_;

Хранит номер читательского билета из таблицы Readers\_catalog

public string surname\_;

Хранит фамилию читателя из таблицы Readers\_catalog

public string name\_;

Хранит имя читателя из таблицы Readers\_catalog

public string patronymic\_;

Хранит отчество читателя из таблицы Readers\_catalog

public DateTime birthday\_;

Хранит дату рождения из таблицы Readers\_catalog

public string education\_;

Хранит образование из таблицы Readers\_catalog

public string profession\_;

Хранит профессию из таблицы Readers\_catalog

public string institut;

Хранит место учёбы из таблицы Readers\_catalog

public string city\_;

Хранит город проживания из таблицы Readers\_catalog

public string street\_;

Хранит улицу проживания из таблицы Readers\_catalog

public int house\_;

Хранит номер дома из таблицы Readers\_catalog

public string building;

Хранит корпус дома из таблицы Readers\_catalog

public int flat\_;

Хранит номер квартиры из таблицы Readers\_catalog

public string phone\_;

Хранит номер телефона читателя из таблицы Readers\_catalog

public int series;

Хранит серию паспорта из таблицы Readers\_catalog

public int number;

Хранит номер паспорта из таблицы Readers\_catalog

public string issued;

Хранит кто выдал паспорт из таблицы Readers\_catalog

public DateTime date;

Хранит дату выдачи паспорта из таблицы Readers\_catalog

public DateTime consists;

Хранит дату, с которой стал читателем библиотеки из таблицы Readers\_catalog

public DateTime re;

Хранит дату перерегистрации из таблицы Readers\_catalog

Конструктор

public Reader(int id\_, string surname\_, string name\_, string patronymic\_, DateTime birthday\_, string education\_, string profession\_, string institut, string city\_, string street\_, int house\_, string building, int flat\_, string phone\_, int series, int number, string issued, DateTime date, DateTime consists, DateTime re)

Параметры

int id\_

string surname\_

string name\_

string patronymic\_

DateTime birthday\_

string education\_

string profession\_

string institute

string city\_

string street\_

int house\_

string building

int flat\_

string phone\_

int series

int number

string issued

DateTime date

DateTime consists

DateTime re

Класс User

public class User

Поля

public int id\_;

Хранит идентификатор пользователя

public string name;

Хранит логин пользователя

public string password\_;

Хранит пароль пользователя

Конструктор

public User(int id\_, string name\_, string password\_)

Параметры

int id\_

string name\_

string password\_

Класс Author\_book

public class Author\_book

Поля

public int id;

Хранит идентификатор записи из таблицы Authors\_Books

public int book\_id;

Хранит идекс книги из таблицы Authors\_Books

public int author\_id;

Хранит идентификатор автора из таблицы Authors\_Books

Конструктор

public Author\_book(int id, int author\_id, int book\_id)

Параметры

int id

int author\_id

int book\_id

Класс BookDebtorsJob

public class BookDebtorsJob

Метод

public async Task Execute(IJobExecutionContext context)

Параметр

IJobExecutionContext context

Экземпляр класса IJobExecutionContext, который предоставляет информацию о текущем выполнении задания

Назначение

Создаёт сообщение о должниках

Возвращаемое значение

Task

Возвращает сообщение, в котором содержится информация о должниках

Класс Scheduler

public class Scheduler

Метод

public static async Task Start()

Назначение

Выполнять задачи в фоновом режиме по расписанию

Возвращаемое значение

Task

Возвращает задачу, которая выполняется в фоновом режиме

Класс PasswordHasher

public class PasswordHasher

Методы

public string HashPassword(string password)

Параметр

string password

Строка, содержащая пароль

Назначение

Хеширует пароль

Возвращаемое значение

string

Возвращает строку с хешированным паролем

public bool VerifyPassword(string password, string hashedPassword)

Параметры

string password

Строка, содержащая пароль

string hashedPassword

Строка, содержащая хешированный пароль

Назначение

Сравнивает пароль с хешированным паролем

Возвращаемое значение

bool

Возвращает true, если пароль совпал с хешированным паролем, false – если не совпал

Класс Authors

public class Authors

Поля

public List<Author> authors = new List<Author>();

Список экземпляров класса Author

string connectionString = "Host = localhost; Username = postgres; Password = 1234; Database = newDB";

Строка для подключения к базе данных

Методы

public void Add(Author a)

Параметр

Author a

Экземпляр класса Author

Назначение

Добавляет экземпляр класса Author в список

public void Del(int id)

Параметр

int id

Идентификатор читателя из таблицы Authors\_catalog

Назначение

Удаляет из списка экземпляр класса Author со значением поля id\_ = id

public Author Find(int id)

Параметр

int id

Идентификатор читателя из таблицы Authors\_catalog

Назначение

Находит экземпляр класса Author в списке

Возвращаемое значение

Author

Экземпляр класса Author из списка

public int FindMaxId()

Назначение

Находит максимальное значение поля id\_ экземпляра класса Author

Возвращаемое значение

int

Возвращает значение максимального идентификатора

public void Add()

Назначение

Заполняет список экземплярами класса Author, значения для которого получает из таблицы Authors\_catalog

public void DelDb(int index)

Параметр

int index

Идентификатор автора из таблицы Authors\_catalog

Назначение

Удаляет из базы данных строку, id которой = index, а также удаляет из списка экземпляр класса Author, значение поля id\_ которого = index

public void UpdDb(string set, int index)

Параметры

string set

Строка, содержащая названия полей таблицы Authors\_catalog и их новые значения

int index

Идентификатор автора из таблицы Authors\_catalog

Назначение

Обновляет строку таблицы Authors\_catalog

public void AddDb(Author author)

Параметры

Author author

Экземпляр класса Author

Назначение

Добавляет новую строку в таблицу Authors\_catalog

Класс Authors\_books

public class Authors\_books

Поля

public List<Author\_book> a\_b = new List<Author\_book>();

Список экземпляров класса Author\_book

string connectionString = "Host = localhost; Username = postgres; Password = 1234; Database = newDB";

Строка для подключения к базе данных

Методы

public void Add(Author\_book a)

Параметр

Author\_book a

Экземпляр класса Author\_book

Назначение

Добавляет экземпляр класса Author\_book в список

public void Del(int id)

Параметр

int id

Идентификатор записи из таблицы Authors\_Books

Назначение

Удаляет из списка экземпляр класса Author\_book со значением поля id\_ = id

public void Del(int id, int index)

Параметры

int id

Идентификатор автора из таблицы Authors\_Books

int index

Индекс книги из таблицы Authors\_Books

Назначение

Удаляет из списка экземпляр класса Author\_book со значением полей author\_id = id и book\_id = index

public Author\_book Find(int id)

Параметр

int id

Идентификатор записи из таблицы Authors\_Books

Назначение

Находит экземпляр класса Author\_book в списке

Возвращаемое значение

Author\_book

Экземпляр класса Author\_book из списка

public bool Find(int id, int index)

Параметры

int id

Идентификатор читателя из таблицы Authors\_Books

int index

Индекс книги из таблицы Authors\_Books

Назначение

Находит экземпляр класса Author\_book в списке

Возвращаемое значение

bool

Возвращает true, если экземпляр класса Author\_book найден, false – если не найден

public Author\_book FindII(int id, int index)

Параметр

int id

Идентификатор читателя из таблицы Authors\_Books

int index

Индекс книги из таблицы Authors\_Books

Назначение

Находит экземпляр класса Author\_book в списке

Возвращаемое значение

Author\_book

Экземпляр класса Author\_book из списка

public int FindMaxId()

Назначение

Находит максимальное значение поля id\_ экземпляра класса Author\_book

Возвращаемое значение

int

Возвращает значение максимального идентификатора

public void Add()

Назначение

Заполняет список экземплярами класса Author\_book, значения для которого получает из таблицы Authors\_Books

public void DelDb(int index)

Параметр

int index

Идентификатор автора из таблицы Authors\_Books

Назначение

Удаляет из базы данных строку, id которой = index, а также удаляет из списка экземпляр класса Author\_book, значение поля id\_ которого = index

public void UpdDb(string set, int index)

Параметры

string set

Строка, содержащая названия полей таблицы Authors\_Books и их новые значения

int index

Идентификатор автора из таблицы Authors\_Books

Назначение

Обновляет строку таблицы Authors\_Books

public void AddDb(Author\_book author)

Параметры

Author\_book author

Экземпляр класса Author\_book

Назначение

Добавляет новую строку в таблицу Authors\_Books

Класс Bookings

public class Bookings

Поля

public List<Booking> bookings = new List<Booking>();

Список экземпляров класса Booking

string connectionString = "Host = localhost; Username = postgres; Password = 1234; Database = newDB";

Строка для подключения к базе данных

Методы

public void Add(Booking a)

Параметр

Booking a

Экземпляр класса Booking

Назначение

Добавляет экземпляр класса Booking в список

public void Del(int id)

Параметр

int id

Номер брони из таблицы Booking\_catalog

Назначение

Удаляет из списка экземпляр класса Booking со значением поля id\_ = id

public Booking Find(int id)

Параметр

int id

Номер брони из таблицы Booking\_catalog

Назначение

Находит экземпляр класса Booking в списке

Возвращаемое значение

Booking

Экземпляр класса Booking из списка

public bool FindReader(int reader)

Параметры

int reader

Идентификатор читателя из таблицы Booking\_catalog

Назначение

Находит экземпляр класса Booking в списке

Возвращаемое значение

bool

Возвращает true, если экземпляр класса Booking найден, false – если не найден

public void DelDb(int index)

Параметр

int index

Идентификатор автора из таблицы Booking\_catalog

Назначение

Удаляет из базы данных строку, id которой = index, а также удаляет из списка экземпляр класса Booking, значение поля id\_ которого = index

public void UpdDb(string set, int index)

Параметры

string set

Строка, содержащая названия полей таблицы Booking\_catalog и их новые значения

int index

Идентификатор записи из таблицы Booking\_catalog

Назначение

Обновляет строку таблицы Booking\_catalog

public void AddDb(Booking booking)

Параметры

Booking booking

Экземпляр класса Booking

Назначение

Добавляет новую строку в таблицу Booking\_catalog

Класс Books

public class Books

Поля

public List<Book> authors = new List<Book>();

Список экземпляров класса Book

string connectionString = "Host = localhost; Username = postgres; Password = 1234; Database = newDB";

Строка для подключения к базе данных

Методы

public void Add(Book a)

Параметр

Book a

Экземпляр класса Book

Назначение

Добавляет экземпляр класса Book в список

public void Del(int id)

Параметр

int id

Индекс книги из таблицы Books\_catalog

Назначение

Удаляет из списка экземпляр класса Book со значением поля index\_ = id

public Book Find(int id)

Параметр

int id

Индекс книги из таблицы Books\_catalog

Назначение

Находит экземпляр класса Book в списке

Возвращаемое значение

Book

Экземпляр класса Book из списка

public void Add()

Назначение

Заполняет список экземплярами класса Book, значения для которого получает из таблицы Books\_catalog

public void DelDb(int index)

Параметр

int index

Идентификатор автора из таблицы Books\_catalog

Назначение

Удаляет из базы данных строку, index которой = index, а также удаляет из списка экземпляр класса Book, значение поля index\_ которого = index

public void UpdDb(string set, int index)

Параметры

string set

Строка, содержащая названия полей таблицы Books\_catalog и их новые значения

int index

Идентификатор автора из таблицы Books\_catalog

Назначение

Обновляет строку таблицы Books\_catalog

public void AddDb(Book b)

Параметры

Book b

Экземпляр класса Book

Назначение

Добавляет новую строку в таблицу Books\_catalog

Класс Orders

public class Orders

Поля

public List<Order> bookings = new List<Order>();

Список экземпляров класса Order

string connectionString = "Host = localhost; Username = postgres; Password = 1234; Database = newDB";

Строка для подключения к базе данных

Методы

public void Add(Order a)

Параметр

Order a

Экземпляр класса Order

Назначение

Добавляет экземпляр класса Order в список

public void Del(int id)

Параметр

int id

Номер заказа из таблицы Order\_catalog

Назначение

Удаляет из списка экземпляр класса Order со значением поля id\_ = id

public Order Find(int id)

Параметр

int id

Номер заказа из таблицы Order\_catalog

Назначение

Находит экземпляр класса Order в списке

Возвращаемое значение

Order

Экземпляр класса Order из списка

public bool FindReader(int reader)

Параметры

int reader

Идентификатор читателя из таблицы Order\_catalog

Назначение

Находит экземпляр класса Order в списке

Возвращаемое значение

bool

Возвращает true, если экземпляр класса Order найден, false – если не найден

public void DelDb(int index)

Параметр

int index

Идентификатор автора из таблицы Order\_catalog

Назначение

Удаляет из базы данных строку, id которой = index, а также удаляет из списка экземпляр класса Order, значение поля id\_ которого = index

public void UpdDb(string set, int index)

Параметры

string set

Строка, содержащая названия полей таблицы Order\_catalog и их новые значения

int index

Идентификатор записи из таблицы Order\_catalog

Назначение

Обновляет строку таблицы Order\_catalog

public void AddDb(Order o)

Параметры

Order o

Экземпляр класса Order

Назначение

Добавляет новую строку в таблицу Order\_catalog

Класс Readers

public class Readers

Поля

public List< Readers> authors = new List< Readers>();

Список экземпляров класса Reader

string connectionString = "Host = localhost; Username = postgres; Password = 1234; Database = newDB";

Строка для подключения к базе данных

Методы

public void Add(Reader a)

Параметр

Reader a

Экземпляр класса Reader

Назначение

Добавляет экземпляр класса Reader в список

public void Del(int id)

Параметр

int id

Идентификатор читателя из таблицы Readers\_catalog

Назначение

Удаляет из списка экземпляр класса Reader со значением поля id\_ = id

public Reader Find(int id)

Параметр

int id

Идентификатор читателя из таблицы Readers\_catalog

Назначение

Находит экземпляр класса Reader в списке

Возвращаемое значение

Readers

Экземпляр класса Reader из списка

public int FindMaxId()

Назначение

Находит максимальное значение поля id\_ экземпляра класса Reader

Возвращаемое значение

int

Возвращает значение максимального идентификатора

public void Add()

Назначение

Заполняет список экземплярами класса Reader, значения для которого получает из таблицы Readers\_catalog

public void DelDb(int index)

Параметр

int index

Идентификатор автора из таблицы Readers\_catalog

Назначение

Удаляет из базы данных строку, id которой = index, а также удаляет из списка экземпляр класса Reader, значение поля id\_ которого = index

public void UpdDb(string set, int index)

Параметры

string set

Строка, содержащая названия полей таблицы Readers\_catalog и их новые значения

int index

Идентификатор автора из таблицы Readers\_catalog

Назначение

Обновляет строку таблицы Readers\_catalog

public void AddDb(Readers a)

Параметры

Reader a

Экземпляр класса Reader

Назначение

Добавляет новую строку в таблицу Readers\_catalog

**3.4 Инструкция пользователя**

Данная программа предназначена для изменения данных о читателях и библиотечном фонде. В ней можно просматривать, изменять, добавлять и удалять информацию из библиотечных каталогов.

При запуске программы появляется первое окно «Меню» (рисунок 2), в котором присутствуют 5 кнопок.

* чтобы попасть в окно с информацией о читателях, необходимо нажать на кнопку «Читатели»;
* чтобы попасть в окно с информацией о бронировании, необходимо нажать на кнопку «Бронирование»;
* чтобы попасть в окно с информацией о заказах, необходимо нажать на кнопку «Заказы»;
* чтобы попасть в окно с информацией о книгах, необходимо нажать на кнопку «Книги»;
* чтобы попасть в окно с информацией об авторах, необходимо нажать на кнопку «Авторы».

В окне «Читатели» (рисунок 3) представлена информация о читателях в таблице. Внизу окна есть 5 кнопок.

* чтобы редактировать запись (строку с информацией о читателе) необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Редактировать», чтобы появилось окно «Читатель» с информацией о читателе;
* чтобы добавить читателя необходимо нажать на кнопку «Добавить», чтобы появилось окно «Читатель» с пустыми полями;
* чтобы удалить читателя необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Удалить». После этого появится окошечко с подтверждением действия. Нажмите на кнопку «Да», чтобы подтвердить удаление, на кнопку «Нет» – чтобы отменить. При нажатии на кнопку «Нет» появится новое окошечко, которое сообщает, что удаление отменено. В этом окошечке необходимо нажать на кнопку «Ок»;
* чтобы посмотреть брони читателя необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Бронь», чтобы появилось окно «Бронирование» с бронями только этого читателя;
* чтобы посмотреть заказы читателя необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Заказ», чтобы появилось окно «Заказы» с заказами только этого читателя.

В окне «Читатель» (рисунок 4) есть текстовые поля, в которых можно изменять необходимую информацию о читателе. Также есть 2 кнопки внизу окна.

* чтобы изменить необходимую информацию о читателе выберите нужное текстовое поле (каждое текстовое поле подписано сверху);
* чтобы сохранить эти изменения необходимо нажать на кнопку «Сохранить изменения». После этого окно закроется;
* кнопка «Пароль» предназначена только для тех читателей, которые хотят пользоваться сайтом. Эта кнопка сгенерирует логин читателя и покажет его в отдельном окошечке. После нажатия кнопки «Ок» откроется окно «Пароль».

В окне «Пароль» (рисунок 5) есть текстовое поле, в которое вводится пароль читателя. После ввода пароля, чтобы сохранить изменения, необходимо нажать на кнопку «Сохранить изменения»

В окне «Бронирование» (рисунок 6) представлена информация о бронированиях в таблице. Внизу окна есть 4 кнопки.

* чтобы редактировать запись (строку с информацией о брони) необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Редактировать», чтобы появилось окно «Бронь» с информацией о брони;
* чтобы добавить бронь необходимо нажать на кнопку «Добавить», чтобы появилось окно «Бронь» с пустыми полями;
* чтобы удалить бронь необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Удалить». После этого появится окошечко с подтверждением действия. Нажмите на кнопку «Да», чтобы подтвердить удаление, на кнопку «Нет» – чтобы отменить. При нажатии на кнопку «Нет» появится новое окошечко, которое сообщает, что удаление отменено. В этом окошечке необходимо нажать на кнопку «Ок»;
* чтобы посмотреть всю информацию о читателе, который создал это бронирование, необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Читатель», чтобы появилось окно «Читатели» с информацией об этом читателе.

В окне «Бронь» (рисунок 7) есть текстовые поля, в которых можно изменять необходимую информацию о брони. Также есть 3 кнопки внизу окна и два поля для галочек.

* чтобы изменить необходимую информацию о брони выберите нужное текстовое поле (каждое текстовое поле подписано сверху). Для удобства два поля «ФИО читателя» и «Название книги» оснащены выпадающими списками;
* поля с галочками нужны, чтобы подтвердить информацию о получении книги читателем и о возврате книги;
* чтобы сохранить эти изменения необходимо нажать на кнопку «Сохранить изменения». После этого окно закроется;
* кнопки «Читатели» и «Книги» открывают окна «Читатели» и «Книги» соответственно. Они нужны для того, чтобы посмотреть дополнительную информацию о читателях и книгах.

В окне «Книги» (рисунок 8) представлена информация о книгах в таблице. Внизу окна есть 4 кнопки.

* чтобы редактировать запись (строку с информацией о книге) необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Редактировать», чтобы появилось окно «Книга» с информацией о книге;
* чтобы добавить книгу необходимо нажать на кнопку «Добавить», чтобы появилось окно «Книга» с пустыми полями;
* чтобы удалить книгу необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Удалить». После этого появится окошечко с подтверждением действия. Нажмите на кнопку «Да», чтобы подтвердить удаление, на кнопку «Нет» – чтобы отменить. При нажатии на кнопку «Нет» появится новое окошечко, которое сообщает, что удаление отменено. В этом окошечке необходимо нажать на кнопку «Ок»;
* чтобы посмотреть авторов книги необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Авторы», чтобы появилось окно «Авторы» с информацией об авторах.

В окне «Книга» (рисунок 9) есть текстовые поля, в которых можно изменять необходимую информацию о книге. Также есть 1 кнопка внизу окна и рамка, которая подписана как «Обложка».

* чтобы изменить необходимую информацию о книге выберите нужное текстовое поле (каждое текстовое поле подписано сверху);
* «Обложка» нужна, чтобы отображать картинку с обложкой книги. Чтобы поменять картинку или добавить её, необходимо щёлкнуть левой кнопкой мыши внутри рамки. После этого появится окно, в котором можно выбрать нужный файл. После выбора нужного файла нажмите кнопку «Ок». Выбранная картинка отобразится внутри рамки;
* чтобы сохранить эти изменения необходимо нажать на кнопку «Сохранить изменения». После этого окно закроется.

В окне «Авторы» (рисунок 10) представлена информация об авторах в таблице. Внизу окна есть 4 кнопки.

* чтобы редактировать запись (строку с информацией об авторе) необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Редактировать», чтобы появилось окно «Автор» с информацией об авторе;
* чтобы добавить автора необходимо нажать на кнопку «Добавить», чтобы появилось окно «Автор» с пустыми полями;
* чтобы удалить автора необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Удалить». После этого появится окошечко с подтверждением действия. Нажмите на кнопку «Да», чтобы подтвердить удаление, на кнопку «Нет» – чтобы отменить. При нажатии на кнопку «Нет» появится новое окошечко, которое сообщает, что удаление отменено. В этом окошечке необходимо нажать на кнопку «Ок»;
* чтобы посмотреть книги автора необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Книги», чтобы появилось окно «Книги» с информацией о книгах.

В окне «Автор» (рисунок 11) есть текстовые поля, в которых можно изменять необходимую информацию об авторе. Также есть 1 кнопка внизу окна.

* чтобы изменить необходимую информацию об авторе выберите нужное текстовое поле (каждое текстовое поле подписано сверху);
* чтобы сохранить эти изменения необходимо нажать на кнопку «Сохранить изменения». После этого окно закроется.

В окне «Заказы» (рисунок 12) представлена информация о заказах в таблице. Внизу окна есть 4 кнопки.

* чтобы редактировать запись (строку с информацией о заказе) необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Редактировать», чтобы появилось окно «Заказ» с информацией о заказе;
* чтобы добавить заказ необходимо нажать на кнопку «Добавить», чтобы появилось окно «Заказ» с пустыми полями;
* чтобы удалить заказ необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Удалить». После этого появится окошечко с подтверждением действия. Нажмите на кнопку «Да», чтобы подтвердить удаление, на кнопку «Нет» – чтобы отменить. При нажатии на кнопку «Нет» появится новое окошечко, которое сообщает, что удаление отменено. В этом окошечке необходимо нажать на кнопку «Ок»;
* чтобы посмотреть дополнительную информацию о читателе, сделавшего заказ, необходимо нажать на любой элемент нужной строки, далее нажать на кнопку «Читатель», чтобы появилось окно «Читатели» с информацией о читателе.

В окне «Заказ» (рисунок 12) есть текстовые поля, в которых можно изменять необходимую информацию об авторе. Также есть 2 кнопки внизу окна.

* чтобы изменить необходимую информацию о заказе выберите нужное текстовое поле (каждое текстовое поле подписано сверху). Для удобства поле «ФИО читателя» имеет выпадающий список читателей;
* чтобы сохранить эти изменения необходимо нажать на кнопку «Сохранить изменения». После этого окно закроется;
* кнопка «Читатели» предназначена для вывода информации о читателях.

**4. Тестирование АИС**

**4.1 Тестирование базы данных**

Для начала необходимо проверить SELECT запросы, чтобы убедиться, что пользователь получает правильные данные. Для этого сравним результаты запросов, посылаемые приложением, и результаты запросов внутри редактора базы данных.

Ниже представлен запрос из программы

command = new NpgsqlCommand("SELECT \* FROM kursovaya.\"Readers\_catalog\"", connection)

На рисунке 13 изображён результат запроса.

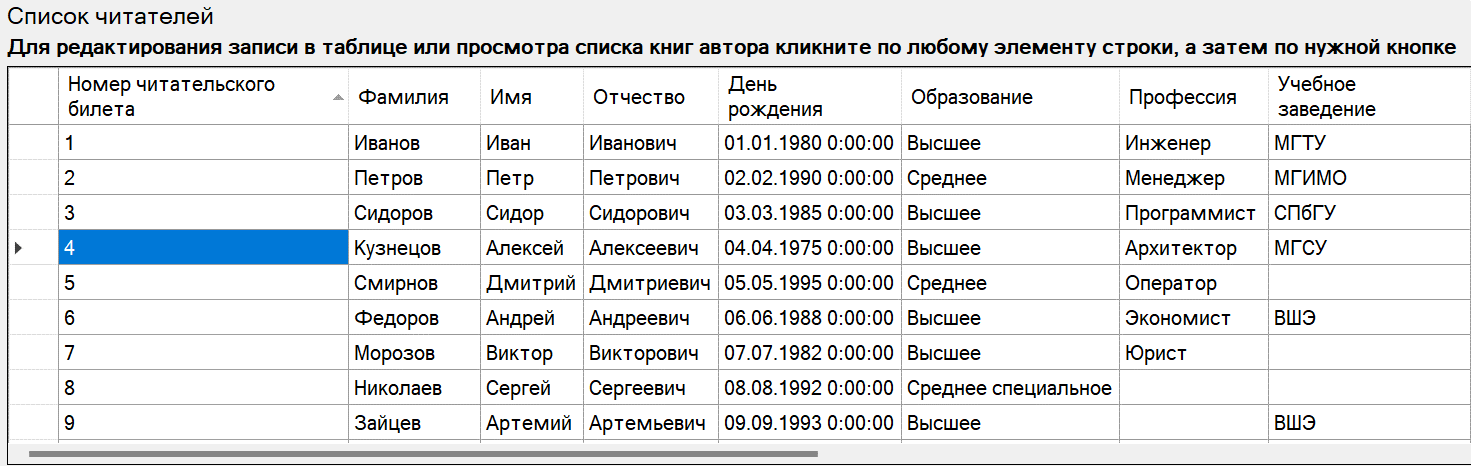


Рисунок 13.

Теперь напишем этот запрос в редакторе.

SELECT \* FROM kursovaya."Readers\_catalog"

На рисунке 14 изображён результат этого запроса.

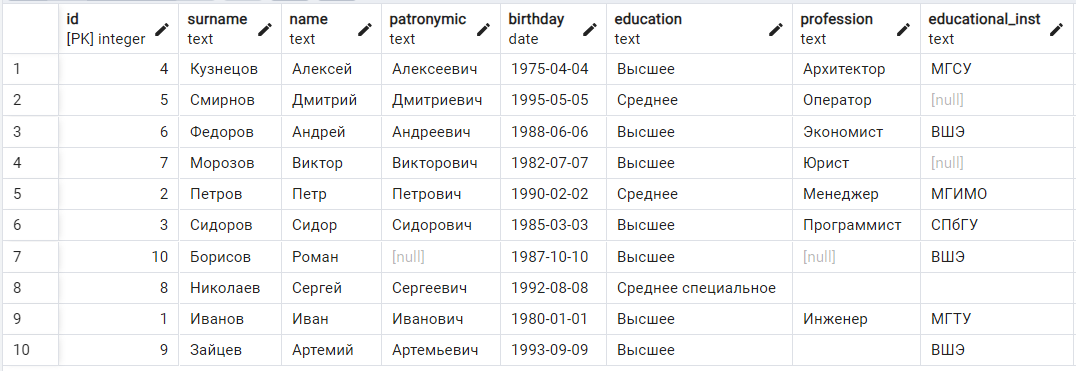


Рисунок 14.

Сравнив поля, можно сделать вывод, что запрос является корректным.

Проведём ещё одно контрольное сравнение, но с другой таблицей.

Запрос из программы

command = new NpgsqlCommand("SELECT \* FROM kursovaya.\"Books\_catalog\"", connection)

Результат на рисунке 15.

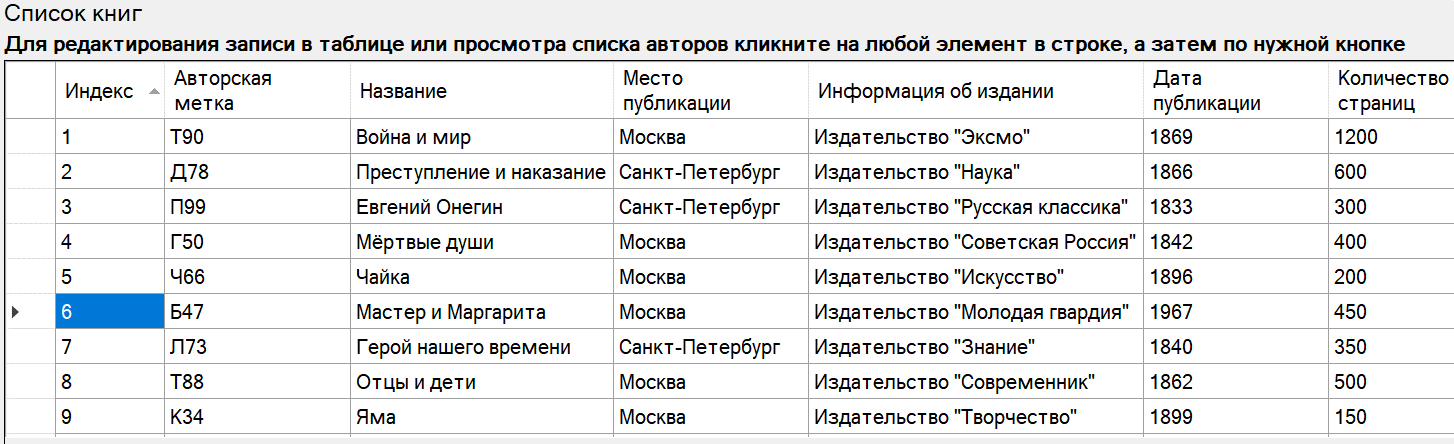


Рисунок 15.

Запрос из редактора

SELECT \* FROM kursovaya."Books\_catalog"

Результат на рисунке 16.

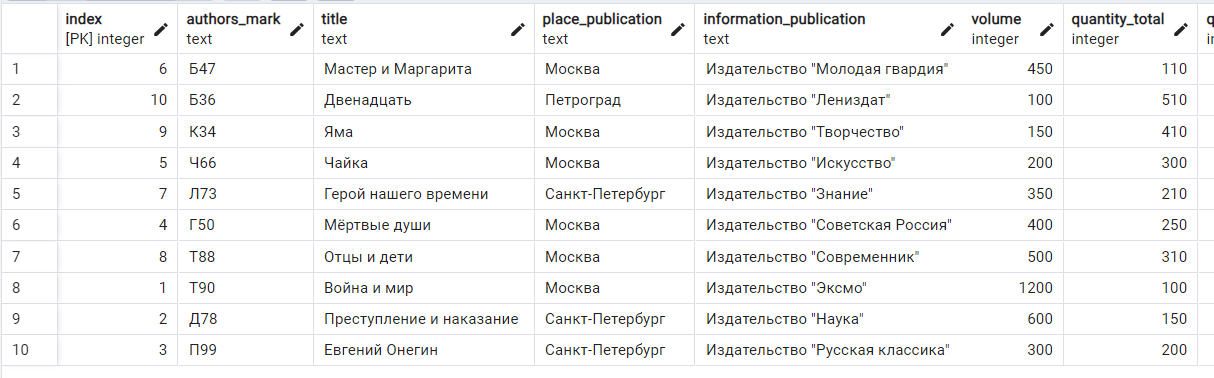


Рисунок 16.

Значения снова совпадают, поэтому делаем вывод, что SELECT запросы корректны

Далее проверим INSERT запросы, чтобы убедиться, что программа корректно создаёт новые записи.

Проверяем следующий запрос из программы

command = new NpgsqlCommand("INSERT INTO kursovaya.\"Authors\_catalog\" VALUES( " + author.id + ", '" +

author.surname + "', '" + author.name + "', '" + author.patronymic + "' )", connection)

Для этого в программе создадим тестового автора (рисунок 17) и проверим его отображение в программе (рисунок 18).

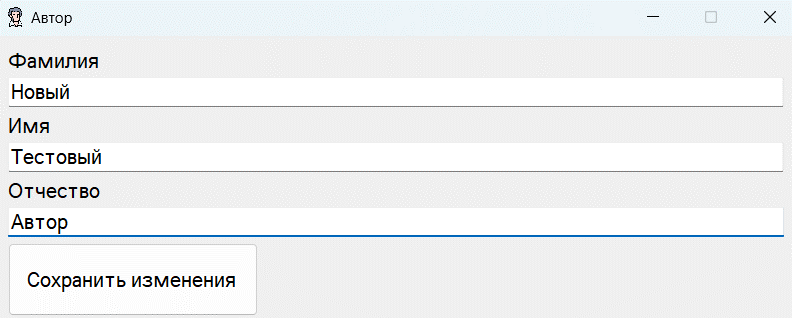


Рисунок 17.

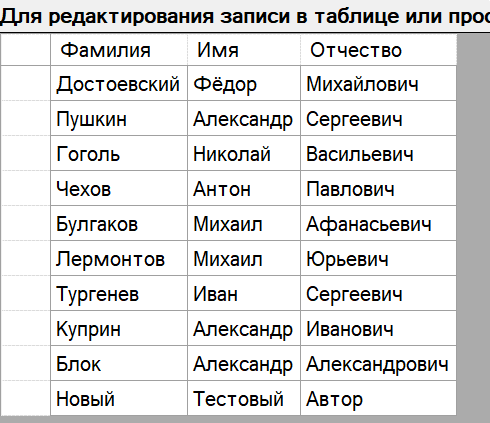


Рисунок 18.

Мы можем видеть, что автор добавился в таблицу в программе. Теперь проверим, есть ли он в базе данных (рисунок 19).



Рисунок 19.

Автор также добавился и в базу данных.

Проверим на второй таблице.

Проверяемый запрос

command = new NpgsqlCommand("INSERT INTO kursovaya.\"Readers\_catalog\" VALUES( " + reader.id\_ + ", '" +

reader.surname\_ + "', '" + reader.name\_ + "', '" + reader.patronymic\_ + "', '" + birthday + "', '" +

reader.education\_ + "', '" + reader.profession\_ + "', '" + reader.institut + "', '" + reader.city\_ + "', '" + reader.street\_ + "', " +

reader.house\_ + ", '" + reader.building + "', " + reader.flat\_ + ", " + reader.series + ", " + reader.number +

", '" + reader.issued + "', '" + date + "', '" + consists + "', '" + re + "', '" + reader.phone\_ + "' )", connection)

Создание пользователя и проверку добавление в программе можно видеть на рисунках 20 и 21 соответственно.

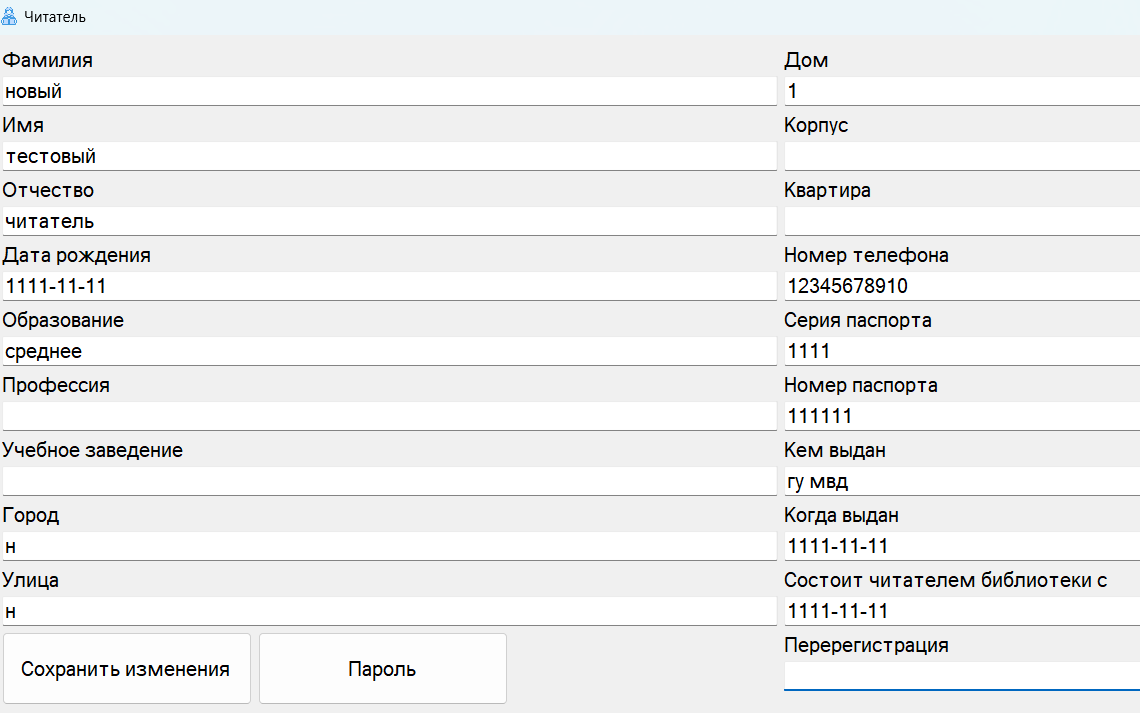


Рисунок 20.

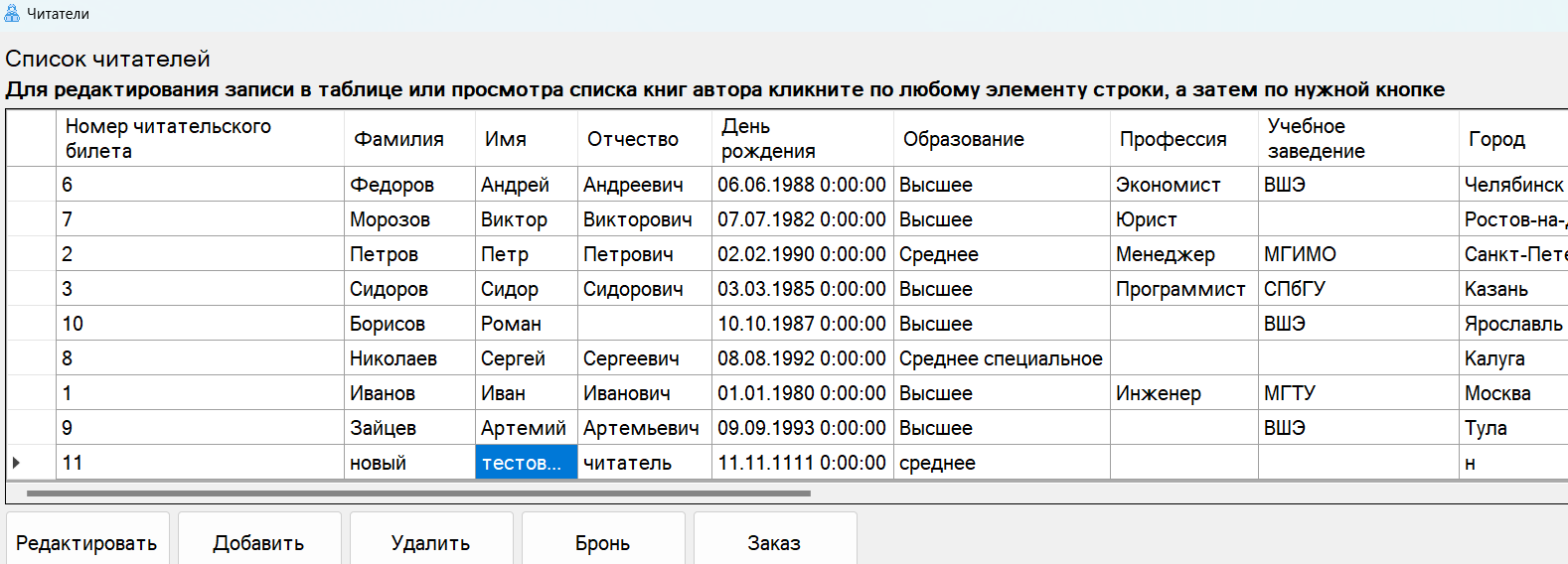


Рисунок 21.

Читатель успешно добавлен в программе. Проверка в базе данных на рисунке 22.

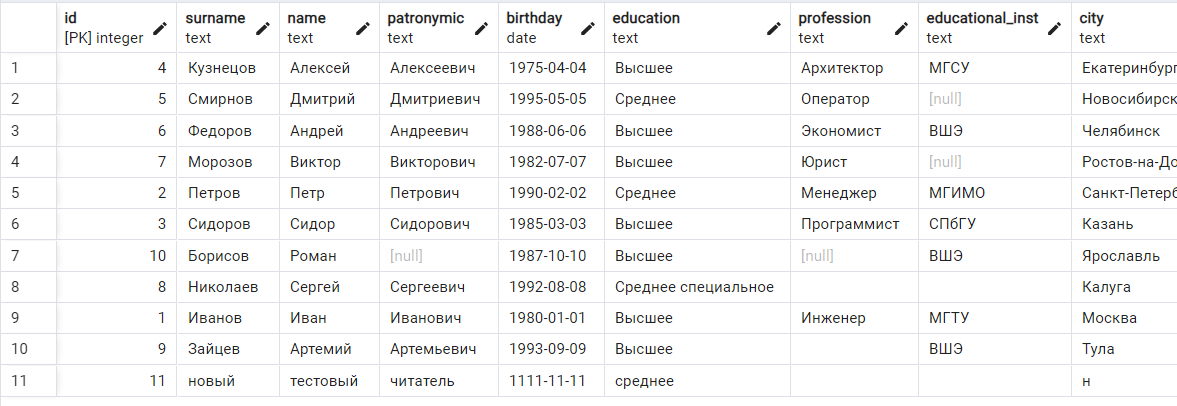


Рисунок 22.

Читатель успешно добавлен в базу данных.

Теперь проверим Update запросы. Для этого поменяем значения полей тестовых записей.

Первый проверяемый запрос

command = new NpgsqlCommand("UPDATE kursovaya.\"Readers\_catalog\" SET " + set + " WHERE id = " + index, connection)

Результат изменения на рисунке 23.



Рисунок 24.

В тестовом читателе было изменено имя и дата рождения. В программе всё работает хорошо.

Сравним с изменениями в базе данных (рисунок 25).



Рисунок 25.

В базе данных также всё успешно.

Проделаем всё то же самое с тестовым автором.

Проверяемый запрос

command = new NpgsqlCommand("SELECT \* FROM kursovaya.\"Authors\_catalog\"", connection)

Результат изменений на рисунке 26.



Рисунок 26.

Изменения затронули имя автора. В программе всё сработало. Изменения в базе данных на рисунке 27.

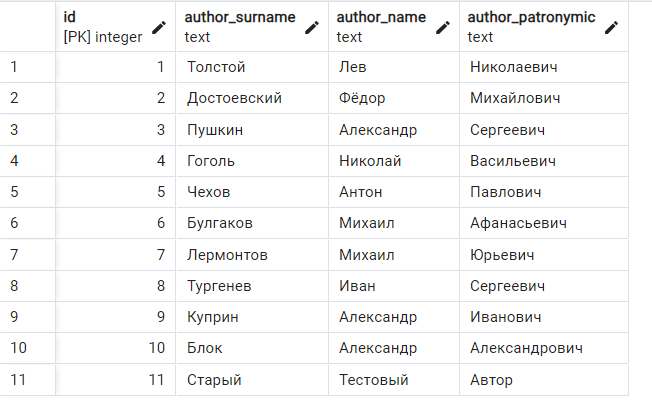


Рисунок 27.

В базе данных всё хорошо.

Теперь проверим DELETE запросы на тестовом авторе и тестовом читателе. Результаты удалений на рисунках 28 и 29 соответственно.

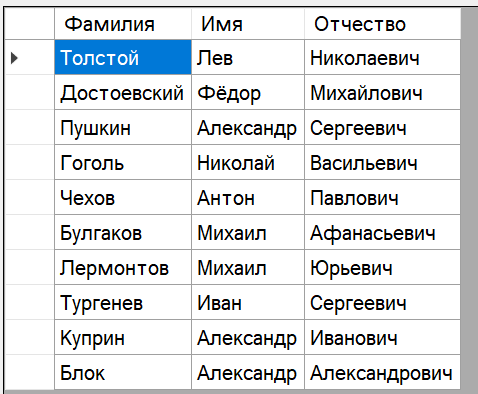


Рисунок 28.

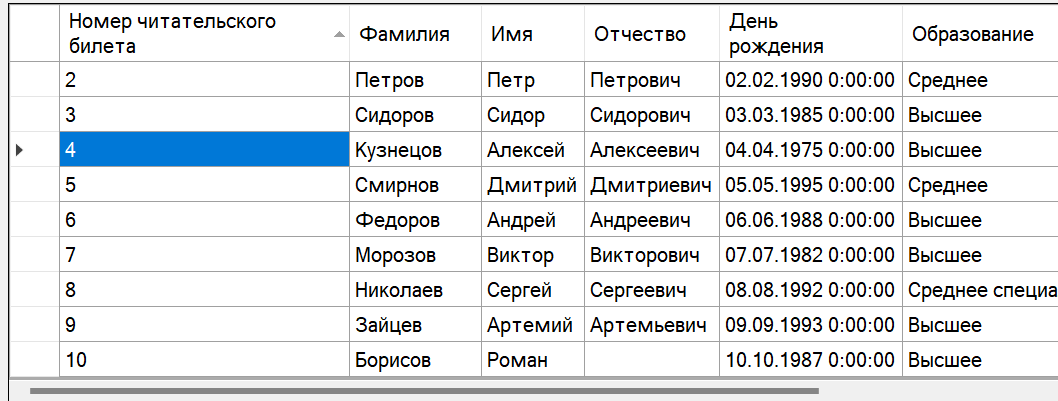


Рисунок 29.

Сравним с результатами в базе данных. Результат для автора на рисунке 30, для читателя – на рисунке 31.



Рисунок 30.

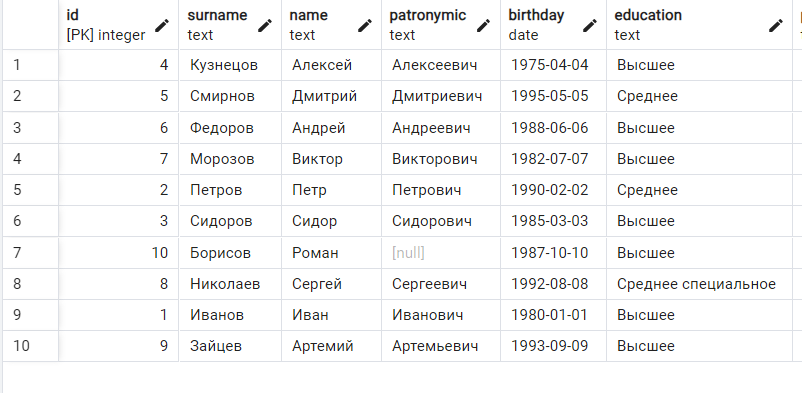


Рисунок 31.

Из результатов видно, что всё работает отлично.

Теперь протестируем триггер и триггерную функцию, описание которых приводилось ранее. Для этого в элементе CheckBox, который отвечает за изменение значения поля returned в таблице Booking\_catalog поставим галочку. Ожидается, что количество книг в библиотеке на данный момент увеличится на значение поля quantity (значение, написанное в тестовом поле «Количество книг») таблицы Booking\_catalog. На рисунке 32 представлен результат изменения элемента CheckBox, на рисунке 33 представлена запись в таблице Books\_catalog до изменения, а на рисунке 34 – после изменения.

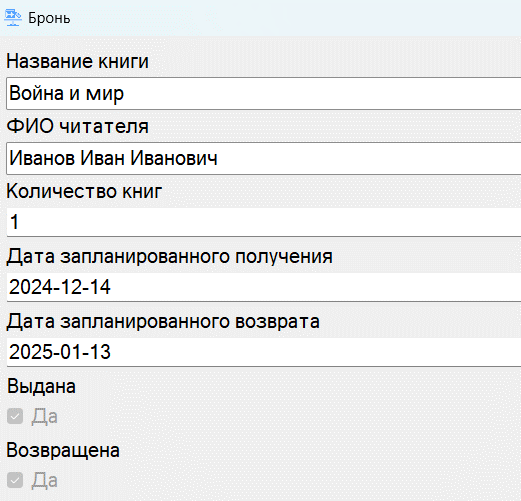


Рисунок 32.



Рисунок 33.



Рисунок 34.

Триггер успешно изменил количество книг. Поэтому можем делать вывод, что запросы к базе данных и сами компоненты базы данных работают корректно.

**4.2 Тестирование программы**

В предыдущем пункте проверялись запросы к базе данных, поэтому все значения вводились корректно. Теперь проверим устойчивость программы, если вводимые значения не будут соответствовать типам полей базы данных.

Попробуем не ввести значение в поле, обязательное для заполнения, то есть передать значение null. Для этого в текстовом поле сотрём значение даты и попытаемся сохранить изменения. Программа выдала тестовое сообщение (рисунок 35). Сработала проверка в следующем фрагменте кода:

else if (string.IsNullOrWhiteSpace(date\_publication.Text))

{

MessageBox.Show("Все текстовые поля должны быть заполнены!");

date\_publication.Focus();

return true;

}

Этот фрагмент проверяет текстовое поле с датой и выдаёт сообщение, что о необходимости в заполнении поля.

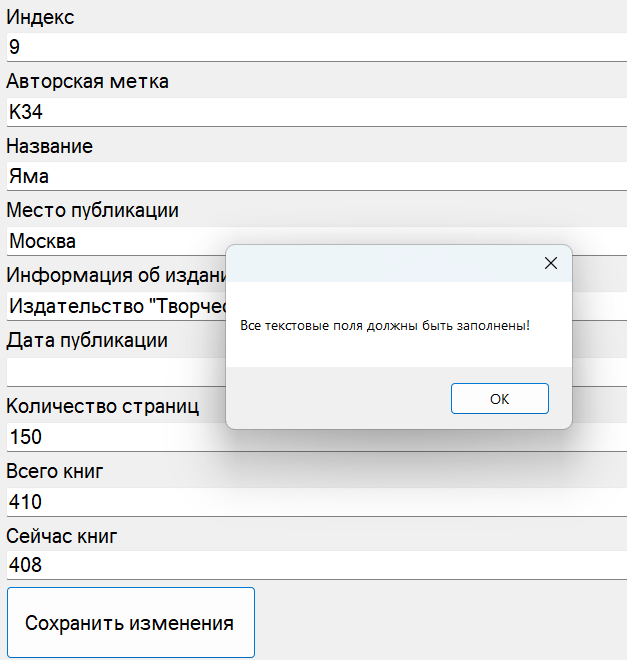


Рисунок 35.

Теперь попробуем ввести значение, которое не существует в базе данных. Для этого в редакторе брони попробуем ввести несуществующее ФИО пользователя (рисунок 36). Программа выдаёт сообщение о необходимости выбора пользователя из базы данных. В данном случае сработал следующий фрагмент кода:

if (fiol[0] != reader.surname\_ || fiol[1] != reader.name\_)

{

Reader reader1 = readers.readers.FirstOrDefault(r => r.surname\_ == fiol[0] && r.name\_ == fiol[1]);

if(reader1 == null) { throw new Exception("Читателя с таким ФИО не существует! Пожалуйста выберите ФИО из списка читателей"); }

set.Add("reader\_id = " + reader1.id\_);

booking.reader = reader1.id\_;

}

В данном фрагменте происходит сравнение предыдущего значения поля с новым значением. Если новое значение не найдено в базе данных, то программа создаёт исключение.

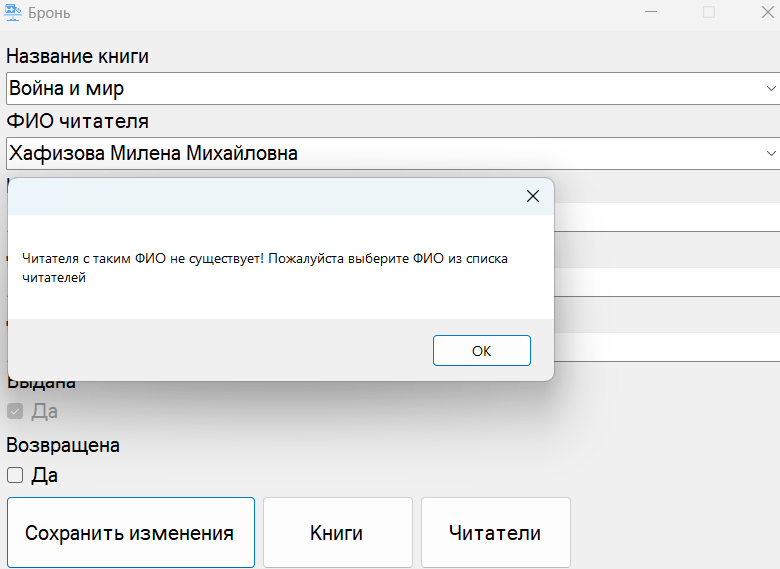


Рисунок 36.

Попробуем при попытке редактировать запись не выбрать запись в компоненте DataGridView. Программа выдаёт сообщение (рисунок 37). Сработал следующий фрагмент кода:

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

{

try

{

if (select == 0)

{

throw new Exception("Строка не выбрана! Пожалуйста нажмите на нужную строку");

}

Book\_form form = new Book\_form(select, books);

form.DataUpdated += DataUpdated;

form.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + "\nДля решения этой проблемы свяжитесь со специалистом");

}

}

При попытке нажать на кнопку программа возвращает исключение, если строка не выбранна.

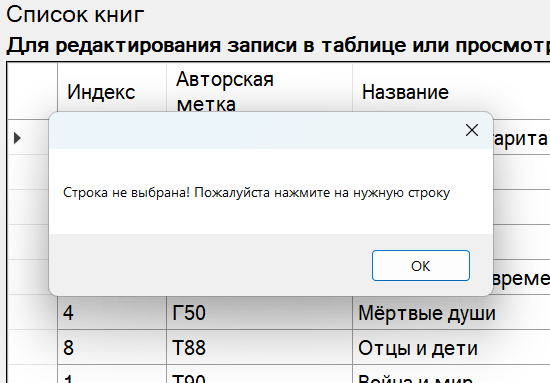


Рисунок 37.

Таким образом тестирование показало, что программа успешно справляется с неправильным форматом данных и ошибочным действием пользователя. Предоставляет развёрнутое сообщение об ошибках.

**Заключение**

В ходе выполнения данной работы была разработана автоматизированная информационная система для библиотеки, которая включает в себя приложение на C# с использованием Windows Forms и интеграцию с базой данных PostgreSQL.

Целью разработки было создание удобного и функционального инструмента для управления библиотечными ресурсами, что может значительно упростить работу библиотекарей и улучшить качество обслуживания пользователей.

Созданное приложение обеспечивает основные функции, такие как учёт книг, регистрацию читателей, управление выдачей и возвратом литературы, а также возможность создания заказов на получение книг, которых нет в библиотеке. Пользовательский интерфейс был разработан с учетом удобства и доступности, что позволяет легко ориентироваться в системе.

Использование PostgreSQL в качестве базы данных обеспечило надежное хранение и обработку информации, что является критически важным для сохранения данных пользователей и ресурсов. В результате реализации проекта были достигнуты все поставленные задачи, и система готова к дальнейшему использованию и доработке.

**Список литературы**

1. Васюткина, И. А. Разработка приложений на C# с использованием СУБД PostgreSQL: учебное пособие : [Текст]/ И. А. Васюткина, Г. В. Трошина, М. И. Бычков, С. А. Менжулин . – Новосибирск: Новосибирский государственный технический университет, 2015. – 143 с.
2. Наместников, А.М. Базы данных. Практический курс. В 2ч. Ч. 1. Объектно-реляционные базы данных на примере PostgreSQL 9.5: учебное пособие / А.М. Наместников, А.А. Филиппов. – Ульяновск: УлГТУ, 2017. – 113c.
3. Самохвалов, Э. Н. Введение в проектирование и разработку приложений на языке программирования С#: учебное пособие / Э. Н. Самохвалов, Г. И. Ревунков, Ю. Е. Гапанюк. – М.: Издательство МГТУ им. Н. Э. Баумана, 2018. – 244 с.
4. О персональных данных: Федеральный закон от 27 июля 2006 года № 152-ФЗ // Собрание законодательства Российской Федерации. – 2006. - № 31. – С. 3451-3457.
5. ГОСТР 7.0.94 – 2022 Система стандартов по информации, библиотечному и издательскому делу – Москва: Российский институт стандартизации, 2022. – 32 с.

**Приложения**

**Приложение А**

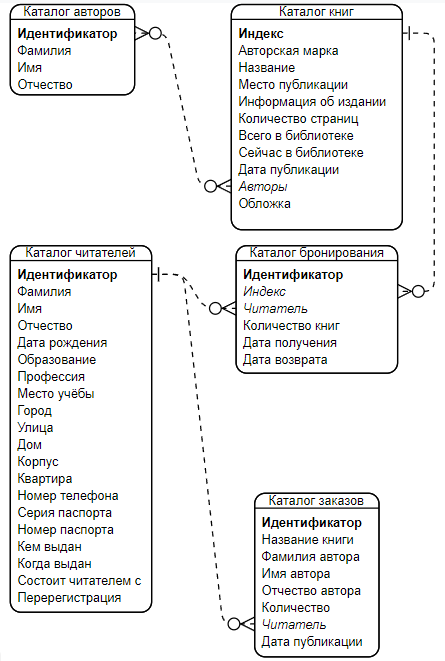


Рисунок А.1 – Логическая модель базы данных.

**Приложение Б**

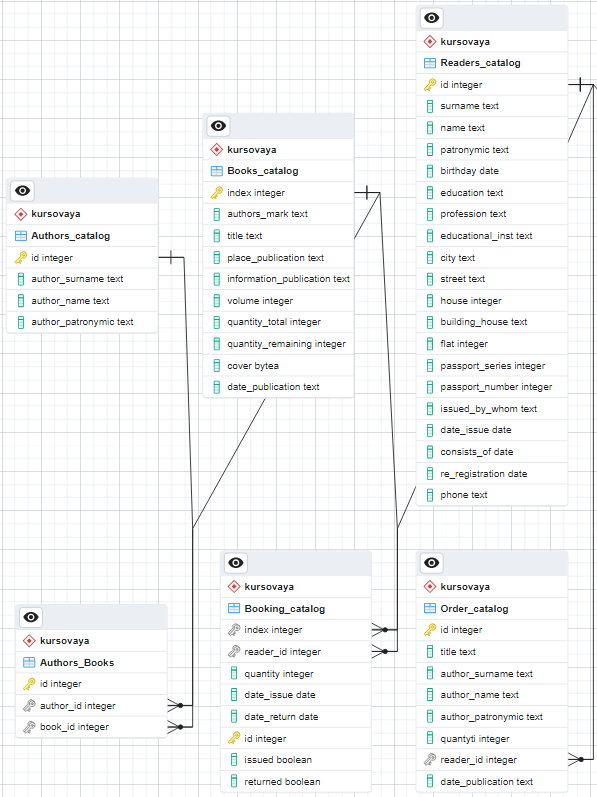


Рисунок Б.1 – Физическая модель базы данных

**Приложение В**

Ссылка на репозиторий с кодом:

<https://github.com/VinskiyL/KursovayaBD_3kurs>

В рамках данной работы была создана АИС библиотеки и модели данных. Система включает в себя базу данных, в которой содержится информация о книгах, их авторах, читателях, записях о получении и возврате книг, о заказах книг в библиотеку. Также создано приложение для библиотекаря, в котором можно работать с данными в таблицах базы данных в графическом интерфейсе.

Работа состоит из вводной части, анализа технического задания, разработки моделей данных, разработки и реализации АИС, тестирования АИС, заключение, списка литературы и приложений.

As part of this work, AIS libraries and data models were created. The system includes a database that contains information about books, their authors, readers, records of receipt and return of books, and orders of books to the library. An application for the librarian has also been created in which you can work with data in database tables in a graphical interface.

The work consists of an introductory part, analysis of the terms of reference, development of data models, development and implementation of AIS, testing of AIS, conclusion, list of references and applications.