МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гомельский государственный университет

имени Франциска Скорины»

Факультет математики и технологий программирования

Кафедра математических проблем управления и информатики

**Разработка приложения для автоматизации работы библиотеки**

Курсовая работа

Исполнитель

студент группы ПИ-31 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.М. Обухов

Научный руководитель

старший преподаватель кафедры МПУИ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Л.И. Короткевич

Гомель 2023

РЕФЕРАТ

Курсовая работа 42 страниц, 15 таблиц, 7 рисунков, 4 приложения, 6 источников

Ключевые слова:база данных, язык программирования, приложение, визуализация данных, данные, средства разработки, запрос

Цель работы:Изучение языков программирования C# и SQL, их взаимодействие, и составление приложения для выполнения, а так же получение различных навыков при использовании этих языков программирования.

Методология проведения работы: изучение среды Visual Studio 2022 и сервера баз данных MS SQL Server, создание базы данных на сервере, разработка и отладка клиент-серверного приложения баз данных.

Полученные результаты работы: база данных на сервере MS SQL Server,

клиент-серверное приложение для автоматизации работы библиотеки.

Область применения результатов: библиотека.

Практическая значимость работы: разработанное приложение упрощает

работу учета книг и выдачи библиотеки.

Выводы:Созданные функциональности приложения позволяют упрощать работу при взаимодействии пользователя и кода. Основная задача курсовой работы достигнута.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3

1. Назначение разработанного приложения 4

2 Описание средств разработки приложения 5

2.1 Среда разработки Visual Studio 2022 5

2.2 Система Управления Базами Данных Microsoft SQL Server 5

2.3 Среда управления SQL Server Management Studio 6

2.4 Язык программирования SQL 7

2.5 Язык программирования C# 7

3 Структура базы данных разработанного приложения 10

3.1 Состав таблиц базы данных 10

3.2 Структура таблиц базы данных 11

4 Взаимодействие пользователя с разработанным приложением 15

ЗАКЛЮЧЕНИЕ 19

ПРИЛОЖЕНИЕ А 21

Скрипт для создания структуры таблиц базы данных 21

ПРИЛОЖЕНИЕ Б 26

Код C#, выполняющий подключение к СУБД, и выполняющий ряд заданных действий для взаимодействия с СУБД 26

ПРИЛОЖЕНИЕ В 38

Код C#, выполняющий логику регистрации и авторизации 38

ПРИЛОЖЕНИЕ Г 41

Результат проверки на плагиат 41

# ВВЕДЕНИЕ

В библиотеках хранится великое число различного рода литературы, которую необходимо каким-либо оптимальным образом хранить, и знать, где её искать. Также библиотеку посещают читатели – они могут брать оттуда книги, при этом идет учет того, что они взяли.

Целью проекта является упрощение работы библиотеки, а именно: выдача книг клиентам(читателям), их хранение, каталогизация, получение и списание; учет клиентов, сотрудников, помещений; контроль корректности данных.  
Данные возможности актуальны в нынешнее время, учитывая тот факт, что в библиотеке, архиве и читальных залах хранится множество единиц различного рода литературы, информацию о которых необходимо собрать воедино и упорядочить для последующего решения проблем по поиску, выдаче и учету книг, а так же книг, которые выдали читателю.

Задачей курсовой работы является создание приложения на базе языка C# в интегрированной среде разработки Visual Studio, где работа идет с созданием GUI формы с подключением базы данных на языке SQL в MS SQL Server для хранения информации и компонентами ADO.Net для связи программы с базой данных (см. приложение А).

Для достижения вышеописанных целей необходимо решить следующие задачи:

* проанализировать требования;
* реализовать серверное подключение с базой данных;
* разработать клиентскую часть, которая будет интерфейсом пользователя, или сотрудника;
* сделать приложение простым и доступным в использовании, а также настроить ограничения на доступ к данным;
* протестировать разработанное приложение.

Разработка визуальной части приложения будет вестись на MS Visual Studio 2022. Для хранения и получения данных будет подключена СУБД на MS SQL Server

# Назначение разработанного приложения

Приложение представляет собой возможность быстрого и удобного взаимодействия с хранимыми данными. Создано для участников библиотеки (читатели, сотрудники) с целью ведения учета хранимых книг, их поставок и списаний, для быстрого доступа к необходимым книгам, а также хранения информации о выдаче книг читателям с целью несения ответственности читателем за полученную им литературу, принадлежащую учреждению (библиотеке), выдавшего книгу.

Подробная информация о книгах, выдачах и читателях станет доступна пользователю, как только он выберет соответствующий раздел, а так же от его прав доступа к данным

Подробная информация, которая будет упомянута выше, будет включать:

* личные данные читателя;
* личные данные сотрудника(-ков);
* полная информация о книгах: их название, жанр, издательство, инвентарный номер и т.п.;
* выдача книг читателям.

Примечание: информация будет доступна только пользователям, имеющим необходимый уровень доступа.

# Описание средств разработки приложения

## **Среда разработки Visual Studio 2022**

Microsoft Visual Studio  - линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, UWP а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, .NET Core, .NET, MAUI, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

## **Система Управления Базами Данных Microsoft SQL Server**

Microsoft SQL Server - система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов - Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия.

Преимущества СУБД:

* создание клиент-серверных приложений (Многопользовательская работа (совместное обновление данных), уменьшение сетевого трафика);
* создание программ, не чувствительных к изменению структуры данных;
* Наличие в СУБД стандартных, качественно реализованных, алгоритмов обработки данных (сортировка, поиск и др.);
* Автоматическое поддержание целостности данных (автоматическое резервное копирование, контроль ссылочной целостности, поддержка механизма транзакций с возможным откатом транзакций);
* разграничение доступа к данным;
* возможность получения данных по запросам без разработки специальных программ.

Центральным аспектом MS SQL является база данных, которая представляет из себя хранилище данных, организованных определенным способом. Для организации баз данных MS SQL использует реляционную модель.

Реляционная модель – подход к управлению данными с использованием языка и структуры, согласованных с логикой первого порядка, где все данные представлены в виде кортежей, сгруппированных в отношения.

Реляционная модель базы данных:

* каждому типу объекта базы данных соответствует таблица. Строка — экземпляр объекта, столбец – атрибут (поле);
* у столбца уникальное имя в таблице. Возможны одинаковые имена столбцов, расположенных в разных таблицах;
* в таблице может содержаться любое количество строк (возможно, нулевое);
* строки таблицы не имеют определенного порядка. Нельзя выбрать строку по номеру в таблице. Нет “первой”, “последней” и т.д. строк;
* в каждой таблице есть один или несколько столбцов, значения или совокупность значений, которых во всех строках различаются. Этот столбец (столбцы) называется первичным ключом таблицы, и выбор конкретной записи таблицы можно осуществить по значению её первичного ключа;
* столбец (совокупность столбцов) одной таблицы, значения в котором совпадают со значениями столбца, являющегося первичным ключом другой таблицы, называется внешним столбцом. Совместно первичный и внешний ключи создают между таблицами отношения предок/потомок (родительская и дочерняя таблицы).

## **Среда управления SQL Server Management Studio**

SQL Server Management server – интегрированная среда для управления любой инфраструктурой SQL. Используется для настройки, доступа, администрирования, разработки всех компонентов SQL. Является стандартным инструментом для работы с MS SQL. Предоставляет удобный интерфейс пользователю для настройки, мониторинга, администрирования экземпляров SQL server и БД. Также предоставляет возможность создавать запросы и скрипты.

Компоненты среды:

* обозреватель объектов – нужен для просмотра всех объектов и управления ими;
* обозреватель шаблонов – нужен для создания файлов со стандартным текстом, которые используются для ускорения разработки скриптов, запросов, и управления ими;
* средства визуального проектирования – нужны для создания запросов, таблиц и схем баз данных;
* редакторы запросов и текста – нужны для интерактивного написания и откладки скриптов, запросов.

## **Язык программирования SQL**

SQL (Structured Query Language - «язык структурированных запросов») — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. Является, прежде всего, информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. В общем случае SQL (без ряда современных расширений) считается языком программирования неполным по Тьюрингу, но вместе с тем стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений.

На языке SQL описываются наборы данных, помогающие получать ответы на вопросы. При использовании SQL необходимо применять правильный синтаксис. Синтаксис - это набор правил, позволяющих правильно сочетать элементы языка. Синтаксис SQL основан на синтаксисе английского языка и имеет много общих элементов с синтаксисом языка Visual Basic для приложений (VBA).

## **Язык программирования C#**

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

C# уже не молодой язык и как и вся платформа .NET уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 11, которая вышла 8 ноября 2022 года вместе с релизом .NET 7.

C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. Поэтому, если вы знакомы с одним из этих языков, то овладеть C# будет легче.

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей.

Роль платформы .NET

Когда говорят C#, нередко имеют в виду технологии платформы .NET (Windows Forms, WPF, ASP.NET, Xamarin). И, наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире.

Как-то Билл Гейтс сказал, что платформа .NET - это лучшее, что создала компания Microsoft. Возможно, он был прав. Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

Поддержка нескольких языков. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому при определенных условиях мы можем сделать отдельные модули одного приложения на отдельных языках.

Кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент - .NET 7 поддерживается на большинстве современных ОС Windows, MacOS, Linux. Используя различные технологии на платформе .NET, можно разрабатывать приложения на языке C# для самых разных платформ - Windows, MacOS, Linux, Android, iOS, Tizen.

Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.

Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET и Entity Framework Core. Для построения графических приложений с богатым насыщенным интерфейсом - технология WPF и WinUI, для создания более простых графических приложений - Windows Forms. Для разработки кроссплатформенных мобильных и десктопных приложений - Xamarin/MAUI. Для создания веб-сайтов и веб-приложений - ASP.NET и т.д.

К этому стоит добавить активной развивающийся и набирающий популяность Blazor - фреймворк, который работает поверх .NET и который позволяет создавать веб-приложения как на стороне сервера, так и на стороне клиента. А в будущем будет поддерживать создание мобильных приложений и, возможно, десктоп-приложений.

Производительность. Согласно ряду тестов веб-приложения на .NET 7 в ряде категорий сильно опережают веб-приложения, построенные с помощью других технологий. Приложения на .NET 7 в принципе отличаются высокой производительностью.

Также еще следует отметить такую особенность языка C# и фреймворка .NET, как автоматическая сборка мусора. А это значит, что нам в большинстве случаев не придется, в отличие от С++, заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память.

# Структура базы данных разработанного приложения

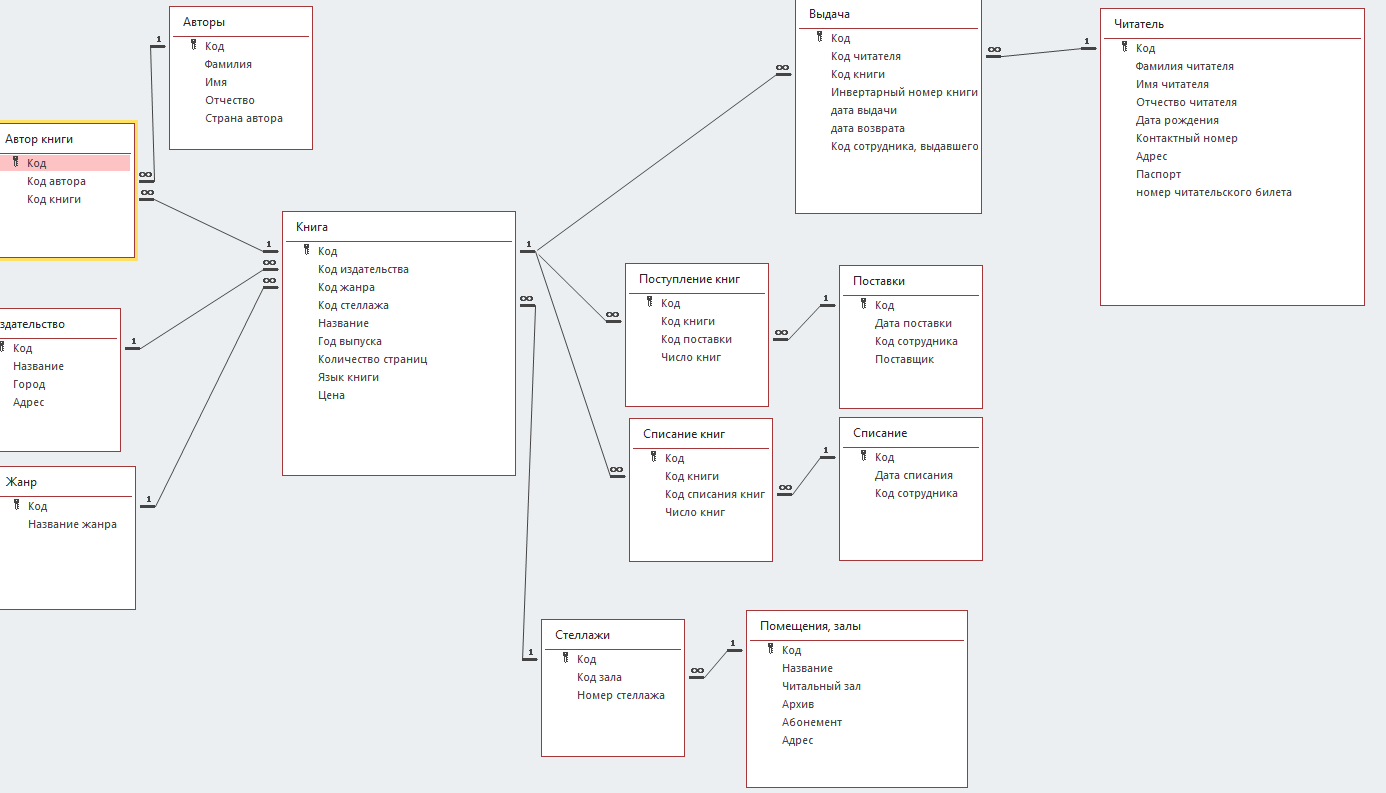
## **Состав таблиц базы данных**

Состав таблиц для базы данных библиотеки представлен в таблице 1.

**Таблица 1 – Состав таблиц базы данных**

|  |  |
| --- | --- |
| Наименование | Общие сведения о содержимом |
| 1. Издательство | издательство, выпустившее книгу |
| 1. Жанр | жанр книги |
| 1. Авторы | список всех авторов с информацией о них |
| 1. Автор книги | служит для связи между таблицами «Авторы», «Книга» |
| 1. Книга | содержит доступную информацию о книге |
| 1. Поставки | информация о поставках книг в библиотеку |
| 1. Списание | информация о списании книг |
| 1. Поступление книг | служит для связи между таблицами «Поставки», «Книга» |
| 1. Списание книг | служит для связи между таблицами «Списание», «Книга» |
| 1. Выдача | информация о выданной литературе читателю |
| 1. Стеллажи | описывает расположение книг на стеллажах |
| 1. Помещения | помещения библиотеки |
| 1. Сотрудники | сотрудники, работающие в библиотеке |
| 1. Читатель | информация о читателях библиотеки |

Взаимосвязь между таблицами указана на рисунке 2.



**Рисунок 1 – Схема базы данных**

## **Структура таблиц базы данных**

В таблицах 2-15 представлена структура таблиц разработанной базы данных.

**Таблица 2 – Издательство**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | код издательства |
| Название | varchar(50) | Название издательства |
| Город | varchar(40) | Город издательства |
| Адрес | varchar(80) | Адрес издательства |

**Таблица 3 – Жанр**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Код жанра |
| Название жанра | varchar(40) | Название жанра книги |

**Таблица 4 – Автор книги**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Код автора книги |
| Код книги | int | Код книги |
| Код автора | int | Код автора, написавшего книгу |

**Таблица 5 – Книга**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Код книги |
| Код издательства | int | Код издательства |
| Код жанра | int | Код жанра |
| Код стеллажа | int | Код стеллажа |
| Название | varchar(50) | Название книги |
| Год выпуска | int | Год выпуска книги |
| Число страниц | int | Число страниц в книге |
| Язык книги | varchar(20) | Язык, на котором издана книга |
| Цена | money | Цена книги при её закупке (в формате BUN) |

**Таблица 6 – Авторы**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Уникальный код Автора |
| Фамилия | varchar(40) | Фамилия автора |
| Имя | varchar(40) | Имя автора |
| Отчество | varchar(40) | Отчество автора |
| Страна автора | varchar(40) | Страна, где родился автор |

**Таблица 7 – Поступление книг**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Уникальный код поступления книг |
| Код книги | int | Код книги |
| Код поставки | int | Код поставки |
| Число книг | int | Число книг в партии |

**Таблица 8 – Поставки**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Уникальный код Поставки |
| Дата поставки | date | Дата, когда были поставлены книги |
| Код сотрудника | int | Код сотрудника, ответственного за получение партии книг |
| Поставщик | varchar(40) | Код поставщика |

**Таблица 9 – Списание книг**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Уникальный код списания книг |
| Код книги | int | Код книги |
| Код списания книг | int | Код списания книг |
| Число книг | int | Число книг, которые списаны за определенный период |

**Таблица 10 – Списание**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Уникальный код Списания |
| Дата списания | date | Дата списания книг |
| Код сотрудника | int | Код сотрудника, ответственного за списание книг |

**Таблица 11 – Выдача**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Уникальный код выдачи |
| Код читателя | int | Код читателя |
| Код книги | int | Код книги |

**Окончание таблицы 11**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Инвентарный номер | varchar(40) | Инвертарный номер книги |
| Дата выдачи | date | Дата выдачи книги читателю |
| Дата возврата | date | Дата возврата книги читателю |
| Код сотрудника, выдавшего книгу | int | Код сотрудника, ответственного за процесс выдачи книги |

**Таблица 12 – Стеллажи**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Уникальный код стеллажа |
| Код зала | int | Код зала, где находится стеллаж |
| Номер стеллажа | varchar(40) | Номер стеллажа (для идентификации) |

**Таблица 13 - Помещения**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Уникальный код помещения |
| Название | varchar(40) | Название помещения |
| Читальный зал | bit | Поле для отметки, чем является данное помещение |
| Архив | bit | Поле для отметки, чем является данное помещение |
| Абонемент | bit | Поле для отметки, чем является данное помещение |
| Адрес помещения | varchar(40) | Адрес помещения |

**Таблица 14 – Читатель**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Уникальный код читателя |
| Фамилия | varchar(40) | Фамилия читателя |
| Имя | varchar(40) | Имя читателя |
| Отчество | varchar(40) | Отчество читателя |

**Окончание таблицы 14**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Дата рождения | date | Дата рождения читателя |
| Контактный номер | varchar(40) | Номер телефона для связи с читателем |
| Адрес проживания | varchar(40) | Адрес, где живет читатель |
| Данные паспорта | char(60) | Идентификационный номер паспорта |
| Номер читательского билета | varchar(40) | Номер читательского билета |

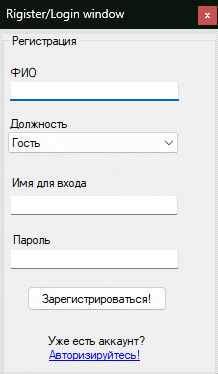
**Таблица 15 - Сотрудники**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Уникальный код сотрудника |
| Код помещения | int | Код помещения, где работает сотрудник |
| Фамилия | varchar(40) | Фамилия сотрудника |
| Имя | varchar(40) | Имя сотрудника |
| Отчество | varchar(40) | Отчество сотрудника |
| Занимаемая должность | varchar(40) | Занимаемая сотрудником должность |
| Стаж | int | Стаж сотрудника (в годах) |

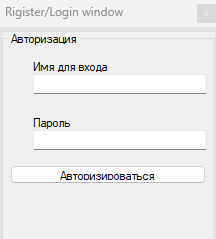
# Взаимодействие пользователя с разработанным приложением

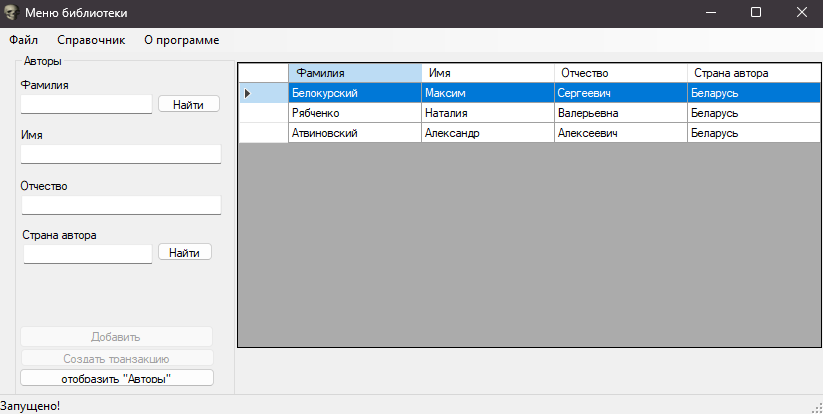
Для управления приложением с подключенной базой данных пользователю предоставляется окно GUI, на котором расположены различные элементы для отображения информации и управлением ею: текстовые поля, поля для ввода текста, выбор даты, списки (в т.ч. выпадающие), и кнопки

Предварительно, пользователь может зарегистрироваться или авторизироваться, чтобы получить доступ к дополнительным возможностям приложения. (По умолчанию активирован базовый функционал: только просмотр книг, авторов, издательств – публичная информация)

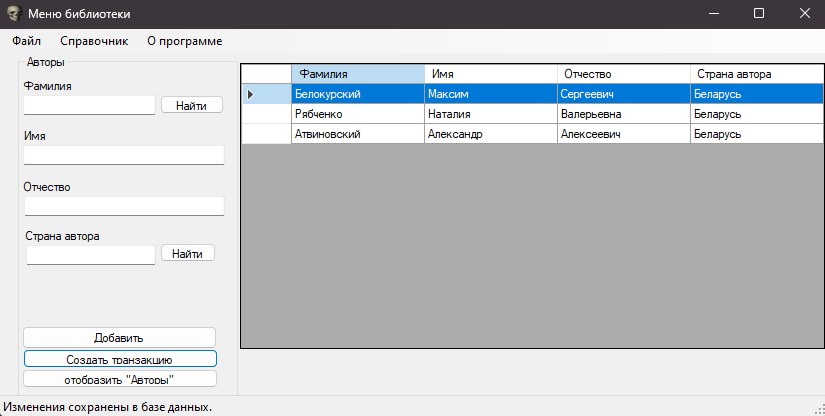


**Рисунок 2 – Регистрационная форма**

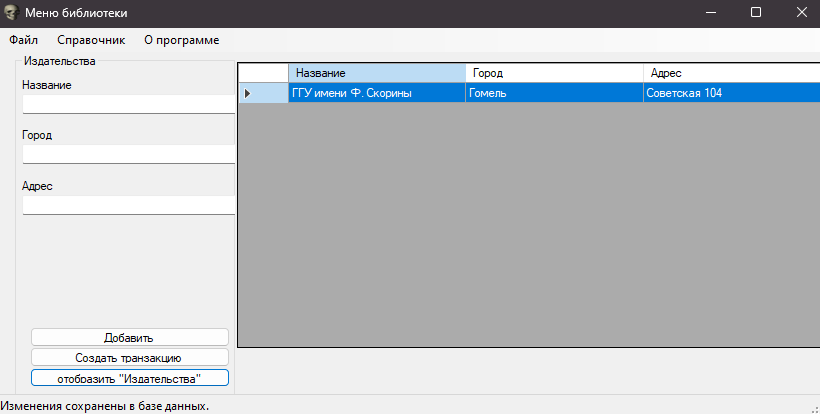
  
**Рисунок 3 – Форма авторизации пользователя**

****

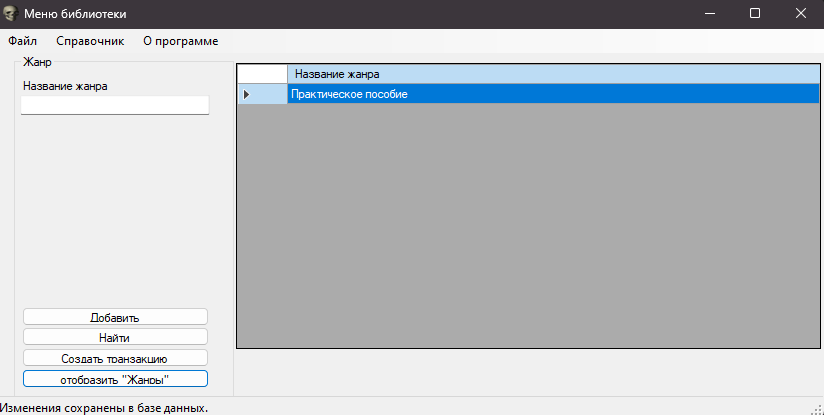
**Рисунок 4 – Рабочая область пользовательского интерфейса для взаимодействия с пользователем (пользователь не вошёл в систему / нет прав). Открыто окно с Авторами книг**

****

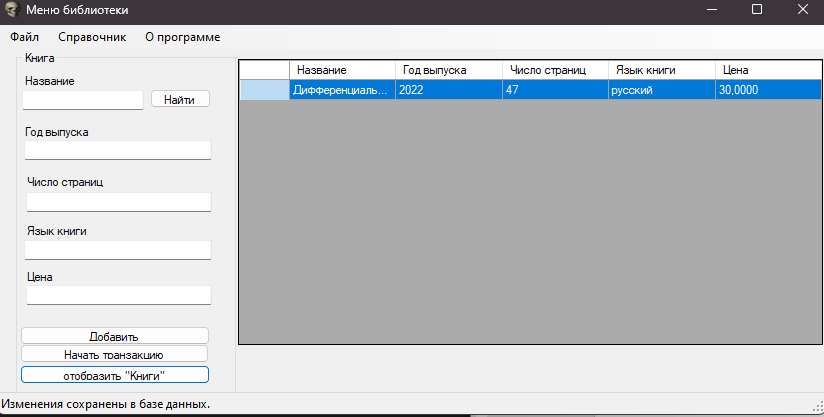
**Рисунок 5 – Рабочая область пользовательского интерфейса для взаимодействия с пользователем (пользователь вошёл в систему, и у него есть права управлять базой данных). Открыто окно с Авторами книг**



**Рисунок 5 – Рабочая область пользовательского интерфейса для взаимодействия с пользователем (пользователь вошёл в систему, и у него есть права управлять базой данных). Открыто окно с Издательствами**

****

**Рисунок 6 – Рабочая область пользовательского интерфейса для взаимодействия с пользователем (пользователь вошёл в систему, и у него есть права управлять базой данных). Открыто окно с Жанры книг**

****

**Рисунок 7 – Рабочая область пользовательского интерфейса для взаимодействия с пользователем (пользователь вошёл в систему, и у него есть права управлять базой данных). Открыто окно с Каталогом книг**

# ЗАКЛЮЧЕНИЕ

Для упрощения работы с людьми и ресурсами библиотеки было создано приложение на языке С#, с подключением СУБД, в которой содержатся различные данные касательно структуры библиотеки, её содержимого, а также посещающих её клиентов (читателей). Пользователям с определёнными правами, таких как сотрудники, разрешено не только просматривать содержимое БД, но и вносить изменения, добавлять, удалять, редактировать информацию в БД.

Для успешного выполнения данной работы были изучены навыки работы с созданием оконных приложений в среде Visual Studio, язык С++, а также реляционные базы данных, в том числе язык запросов SQL.

Для достижения вышеописанных целей были выполнены следующие шаги:

* проанализировать требования;
* реализовать серверное подключение с базой данных;
* разработать клиентскую часть, которая будет интерфейсом пользователя, или сотрудника;
* сделать приложение простым и доступным в использовании, а также настроить ограничения на доступ к данным;
* протестировать разработанное приложение.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шилд Г. C# 4.0. Полное руководство. – М.:Вильямс, 2022. -1056 с.
2. Скит Дж. C# программирование для профессионалов. – М.: Вильямс, 2019. – 608 с.
3. Грин Дж., Стиллмен Э. Изучаем C#/ - СПб.: Питер, 2021, - 816 с.
4. Васильев А.Н. Программирование для начинающих на С#. – М.: Бомбора, 2018, -592 с.
5. Фленов М.Е. Библия C#. – СПб.: БХВ-Петербург, 2020, - 512 с.
6. Лобел Л., Браст Э., Форте С. Разработка приложений на основе Microsoft SQL Server 2008. – СПб.: BHV, 2016. – 1024 с.

# ПРИЛОЖЕНИЕ А

## **Скрипт для создания структуры таблиц базы данных**

alter table Книга

drop constraint FK\_Книга\_Издательство,

constraint FK\_Книга\_Жанр,

constraint FK\_Книга\_Стеллаж

GO

alter table [Автор книги]

drop constraint FK\_Книга\_Автор\_книги,

constraint FK\_Автор\_книги\_Авторы

GO

alter table [Поступление книг]

drop constraint FK\_Книга\_Поступления\_книг,

constraint FK\_Поступление\_книг\_Поставки

GO

alter table [Списание книг]

drop constraint FK\_Книга\_Списание\_книг,

constraint FK\_Списание\_книг\_Списание

GO

alter table [Выдача]

drop constraint FK\_Книга\_Выдача,

constraint FK\_Выдача\_Сотрудники,

constraint FK\_Выдача\_Читатель

GO

alter table [Стеллажи]

drop constraint FK\_Стеллажи\_Помещения

GO

alter table [Сотрудники]

drop constraint FK\_Сотрудники\_Помещения

go

alter table [Поставки]

drop constraint FK\_Поставки\_Сотрудники

go

alter table [Списание]

drop constraint FK\_Списание\_Сотрудники

go

drop table Издательство

go

drop table Жанр

go

drop table Книга

go

drop table [Автор книги]

go

drop table Авторы

go

drop table [Поступление книг]

go

drop table Поставки

go

drop table [Списание книг]

go

drop table Списание

go

drop table Выдача

go

drop table Стеллажи

go

drop table Читатель

go

drop table [Помещения]

go

drop table Сотрудники

go

Create table Издательство

(Код int not null primary key identity,

Название varchar(50),

Город varchar(40),

Адрес varchar(80) not null

)

go

create table Жанр

(Код int primary key not null identity,

[Название жанра] varchar(40) not null

)

GO

create table Книга

(Код int primary key not null identity,

[Код издательства] int,

[Код жанра] int,

[Код стеллажа] int,

[Название] varchar(50),

[Год выпуска] int,

[Число страниц] int,

[Язык книги] varchar(20) not null,

[Цена] money

)

GO

create table [Автор книги]

(Код int primary key not null identity,

[Код книги] int,

[Код автора] int

)

GO

create table Авторы

(Код int primary key not null identity,

[Фамилия] varchar(40),

[Имя] varchar(40),

[Отчество] varchar(40),

[Страна автора] varchar(40)

)

GO

create table [Поступление книг]

(Код int primary key not null identity,

[Код книги] int,

[Код поставки] Int,

[Число книг] int

)

GO

create table Поставки

(Код int primary key not null identity,

[Дата поставки] date,

[Код сотрудника] int,

[Поставщик] varchar(40)

)

GO

create table [Списание книг]

(Код int primary key not null identity,

[Код книги] int,

[Код списания книг] int,

[Число книг] int

)

GO

create table Списание

(Код int primary key not null identity,

[Дата списания] date,

[Код сотрудника] int

)

GO

create table Выдача

(Код int primary key not null identity,

[Код читателя] int,

[Код книги] int,

[Инвертарный номер] varchar(40),

[Дата выдачи] date not null,

[Дата возврата] date,

[Код сотрудника, выдавшего книгу] int

)

GO

create table Стеллажи

(Код int primary key not null identity,

[Код зала] int not null,

[Номер стеллажа] varchar(4)

)

GO

create table Помещения

(Код int primary key not null identity,

[Название] varchar(40),

[Читальный зал] bit,

[Архив] bit,

[Абонемент] bit,

[Адрес помещения] varchar(40)

)

GO

create table Читатель

(Код int primary key not null identity,

[Фамилия] varchar(40),

[Имя] varchar(40),

[Отчество] varchar(40),

[Дата рождения] date,

[Контактный номер] varchar(40),

[Адрес проживания] varchar(40),

[Данные паспорта] char(60),

[Номер читательского билета] varchar(40)

)

GO

create table Сотрудники

(Код int primary key not null identity,

[Код помещения] int,

[Фамилия] varchar(40),

[Имя] varchar(40),

[Отчество] varchar(40),

[Занимаемая должность] varchar(40),

[Стаж] int

)

GO

alter table Книга

add constraint FK\_Книга\_Издательство foreign key ([Код издательства]) references Издательство(Код),

constraint FK\_Книга\_Жанр foreign key ([Код жанра]) references Жанр(Код),

constraint FK\_Книга\_Стеллаж foreign key ([Код стеллажа]) references Стеллажи(Код)

GO

alter table [Автор книги]

add constraint FK\_Книга\_Автор\_книги foreign key ([Код книги]) references Книга(Код),

constraint FK\_Автор\_книги\_Авторы foreign key ([Код автора]) references Авторы(Код)

GO

alter table [Поступление книг]

add constraint FK\_Книга\_Поступления\_книг foreign key ([Код книги]) references Книга(Код),

constraint FK\_Поступление\_книг\_Поставки foreign key ([Код поставки]) references Авторы(Код)

GO

alter table [Списание книг]

add constraint FK\_Книга\_Списание\_книг foreign key ([Код книги]) references Поставки(Код),

constraint FK\_Списание\_книг\_Списание foreign key ([Код списания книг]) references Списание(Код)

GO

alter table [Выдача]

add constraint FK\_Книга\_Выдача foreign key ([Код книги]) references Книга(Код),

constraint FK\_Выдача\_Сотрудники foreign key ([Код сотрудника, выдавшего книгу]) references Сотрудники(Код),

constraint FK\_Выдача\_Читатель foreign key ([Код читателя]) references Читатель(Код)

GO

alter table [Стеллажи]

add constraint FK\_Стеллажи\_Помещения foreign key ([Код зала]) references Помещения(Код)

GO

alter table [Сотрудники]

add constraint FK\_Сотрудники\_Помещения foreign key ([Код помещения]) references Помещения(Код)

GO

alter table [Поставки]

add constraint FK\_Поставки\_Сотрудники foreign key ([Код сотрудника]) references Сотрудники(Код)

GO

alter table [Списание]

add constraint FK\_Списание\_Сотрудники foreign key ([Код сотрудника]) references Сотрудники(Код)

GO

insert into Жанр

values

('Практическое пособие')

insert into Издательство

values

('ГГУ имени Ф. Скорины','Гомель','Советская 104')

insert into Читатель

values

('Обухов','Никита','Михайлович','2002-06-21','+37529#######','Свиридова ##','HB#######','number\_bilet')

insert into Авторы

values

('Белокурский','Максим','Сергеевич','Беларусь'),

('Рябченко','Наталия','Валерьевна','Беларусь'),

('Атвиновский','Александр','Алексеевич','Беларусь')

insert into Помещения(Название,[Адрес помещения])

values

('1й читальный зал при ГГУ им. Ф. Скорины','ул. Советская 102'),

('2й читальный зал при ГГУ им. Ф. Скорины','ул. Советская 102')

insert into Стеллажи([Код зала],[Номер стеллажа])

values

(1,1)

insert into Книга

values

(1,1,1,'Дифференциальные уравнения', 2022, 47, 'русский', 30)

insert into [Автор книги]([Код книги],[Код автора])

values

(1,1),

(1,2),

(1,3)

insert into Сотрудники

values

(1,'Иванов','Петр','Ильич','библиотекарь',3),

(1,'Семченко','Алла','Сергеевна','уборщица',2)

insert into [Поступление книг]([Код книги],[Код поставки],[Число книг])

values

(1,1,10)

insert into [Поставки]

values

('2022-05-07',1,'ГГУ им. Ф. Скорины')

# ПРИЛОЖЕНИЕ Б

## **Код C#, выполняющий подключение к СУБД, и выполняющий ряд заданных действий для взаимодействия с СУБД**

using System;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

private SqlConnection connection = null;

private string[] usersAcessLevels = null;

private static string[] dataSourse = new string[] { "NIKNOTEBOOK", "NIKITPC", "127.0.0.1" };

private DataTable dataTable;

public static string GetDataSources

{

get

{

int usedSorce = 1;

return dataSourse[usedSorce];

}

}

private enum AcсessLevels

{

db\_owner,

db\_securityadmin,

db\_accessadmin,

db\_backupoperator,

db\_ddladmin,

db\_datareader,

db\_datawriter,

db\_denydatareader,

db\_denydatawriter,

db\_executor,

db\_owner\_sid,

db\_securityadmin\_sid,

db\_accessadmin\_sid,

db\_backupoperator\_sid,

db\_ddladmin\_sid,

db\_datareader\_sid,

db\_datawriter\_sid,

db\_denydatareader\_sid,

none

}

public Form1()

{

dataTable = new DataTable();

this.Size = new Size(200, 200);

this.Refresh();

this.AutoSize = true;

this.PerformAutoScale();

InitializeComponent();

//this.MaximizeBox = false;

//this.MinimizeBox = false;

groupBox1.Visible = false;

groupBox2.Visible = false;

groupBox3.Visible = false;

groupBox4.Visible = false;

вызовТаблицыToolStripMenuItem.Visible = false;

dataGridView1.Visible = false;

CheckAcessibility(usersAcessLevels);

dataGridView1.Location = new Point(241, 34);

dataGridView1.Size = new Size(585, 286);

}

private void SetInvisible()

{

foreach (Control control in Controls)

{

if (control is GroupBox groupBox)

{

groupBox.Visible = false;

}

}

}

private void SetInvisible(GroupBox groupBoxToKeepVisible)

{

foreach (Control control in Controls)

{

if (control is GroupBox groupBox)

{

if (groupBox != groupBoxToKeepVisible)

{

groupBox.Visible = false;

}

}

}

}

private void CheckAcessibility(string[] roles)

{

AcсessLevels maxRole = AcсessLevels.none;

if (!(roles is null))

{

foreach (string role in roles)

{

if (Enum.TryParse(role, out AcсessLevels parsedRole) && parsedRole < maxRole)

{

maxRole = parsedRole;

}

}

}

void FunctionalityDisable()

{

add\_author.Enabled = false;

pushAuthor.Enabled = false;

addIzd.Enabled = false;

pushIzd.Enabled = false;

addGenre.Enabled = false;

genreTran.Enabled = false;

addBook.Enabled = false;

pushBook.Enabled = false;

dataGridView1.Enabled = false;

dataGridView1.Visible = false;

вызовТаблицыToolStripMenuItem.Visible = false;

this.MinimumSize = new System.Drawing.Size(260, 48);

}

void FunctionalityEnable()

{

add\_author.Enabled = true;

search\_author.Enabled = true;

lookForCountry.Enabled = true;

pushAuthor.Enabled = true;

addIzd.Enabled = true;

pushIzd.Enabled = true;

addGenre.Enabled = true;

genreTran.Enabled = true;

addBook.Enabled = true;

pushBook.Enabled = true;

//вызовТаблицыToolStripMenuItem.Visible = true;

// this.MinimumSize = new System.Drawing.Size(380, 46);

}

//MessageBox.Show($"{maxRole}", "yes", MessageBoxButtons.OK);

switch (maxRole)

{

case AcсessLevels.db\_owner:

case AcсessLevels.db\_datareader:

case AcсessLevels.db\_datawriter:

FunctionalityEnable();

break;

default:

FunctionalityDisable();

break;

}

if (maxRole.ToString() != "none")

{

statText.Text = $"Авторизирован. Уровень доступа: {maxRole}";

}

}

public void ExecuteQuery(string query)

{

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Console.WriteLine(reader.GetString(0));

}

}

private void doSearchRoutine(string tablename, string field, string qText) {

string que = $"select \* from [{tablename}] where [{field}] like '%{qText}%'";

try

{

using (SqlDataAdapter adapter = new SqlDataAdapter(que, connection))

{

dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

dataGridView1.Enabled = true;

dataGridView1.Visible = true;

dataGridView1.Columns[0].Visible = false;

}

catch (Exception ex) { MessageBox.Show("Ошибка поиска!" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error); }

}

private void DoTransactRoutine(string tablename)

{

if (dataTable != null)

{

SqlTransaction transaction = connection.BeginTransaction();

try

{

using (SqlDataAdapter adapter = new SqlDataAdapter())

{

adapter.SelectCommand = new SqlCommand($"SELECT \* FROM {tablename}", connection);

adapter.SelectCommand.Transaction = transaction;

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);

adapter.UpdateCommand = commandBuilder.GetUpdateCommand();

adapter.InsertCommand = commandBuilder.GetInsertCommand();

adapter.DeleteCommand = commandBuilder.GetDeleteCommand();

adapter.Update(dataTable);

transaction.Commit();

statText.Text = "Изменения сохранены в базе данных.";

}

}

catch (Exception ex)

{

transaction.Rollback();

MessageBox.Show("Произошла ошибка при сохранении изменений: " + ex.Message);

}

}

}

private void DoDisplayRoutine(string tablename)

{

try

{

using (SqlConnection connection = new SqlConnection($"Data Source={GetDataSources};Initial Catalog=Библиотека;Integrated Security=True"))

{

string query = $"SELECT \* FROM {tablename}";

using (SqlDataAdapter adapter = new SqlDataAdapter(query, connection))

{

dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

}

dataGridView1.Enabled = true;

dataGridView1.Visible = true;

dataGridView1.Columns[0].ReadOnly = true;

}

catch (Exception er) { MessageBox.Show("Ошибка отображения! " + er.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error); }

}

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

{

DialogResult = MessageBox.Show("Developer: Никита Обухов\nemail: nikitoniy2468@gmail.com\nAll rights reserved.", "About", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

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

{

SetInvisible();

groupBox1.Visible = true;

groupBox1.Location = new Point(19, 26);

groupBox1.Size = new Size(220, 360);

}

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

{

SetInvisible();

groupBox2.Visible = true;

groupBox2.Location = new Point(19, 26);

groupBox2.Size = new Size(220, 360);

}

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

{

SetInvisible();

groupBox3.Visible = true;

groupBox3.Location = new Point(19, 26);

groupBox3.Size = new Size(220, 360);

}

private void книгаToolStripMenuItem\_Click\_1(object sender, EventArgs e)

{

SetInvisible();

groupBox4.Visible = true;

groupBox4.Location = new Point(19, 26);

groupBox4.Size = new Size(220, 360);

}

private void книгиToolStripMenuItem2\_Click(object sender, EventArgs e)

{

SetInvisible();

groupBox4.Visible = true;

groupBox4.Location = new Point(19, 26);

groupBox4.Size = new Size(220, 360);

}

private void авторыToolStripMenuItem1\_Click(object sender, EventArgs e)

{

SetInvisible();

groupBox1.Visible = true;

groupBox1.Location = new Point(19, 26);

groupBox1.Size = new Size(220, 360);

}

private void жанрыToolStripMenuItem1\_Click(object sender, EventArgs e)

{

SetInvisible();

groupBox3.Visible = true;

groupBox3.Location = new Point(19, 26);

groupBox3.Size = new Size(220, 360);

}

private void издательстваToolStripMenuItem1\_Click(object sender, EventArgs e)

{

SetInvisible();

groupBox2.Visible = true;

groupBox2.Location = new Point(19, 26);

groupBox2.Size = new Size(220, 360);

}

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

{

AuthForm authreg = new AuthForm();

try

{

authreg.WhichWindow("authorize");

authreg.ShowDialog();

connection = authreg.GetConnection;

usersAcessLevels = authreg.GetAcessLevels;

CheckAcessibility(usersAcessLevels);

}

catch { };

}

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

{

AuthForm authForm = new AuthForm();

authForm.WhichWindow("register");

authForm.ShowDialog();

}

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

{

Environment.Exit(0);

}

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

{

connection.Close();

connection = null;

CheckAcessibility(null);

}

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

{

try

{

if (connection != null && connection.State == ConnectionState.Open)

{

string sql = "SELECT SUSER\_SNAME()";

using (SqlCommand command = new SqlCommand(sql, connection))

{

string userName = command.ExecuteScalar().ToString();

MessageBox.Show($"Успешно подключено.\nПользователь: {Environment.UserName};\nКомпьютер: {Environment.MachineName};\nИмя сервера: {connection.DataSource};\nИмя базы данных: {connection.Database};\nВы авторизированы под именем: {userName}", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

else throw new Exception("Вы не авторизированы!");

}

catch (Exception ex) { MessageBox.Show("Невозможно подключиться\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error); }

}

//авторы===========

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

{

string que = "INSERT INTO Авторы (Фамилия, Имя, Отчество, [Страна автора]) VALUES (@Фамилия, @Имя, @Отчество, @Страна)";

using (SqlCommand command = new SqlCommand(que, connection))

{

command.Parameters.AddWithValue("@Фамилия", textBox1.Text);

command.Parameters.AddWithValue("@Имя", textBox2.Text);

command.Parameters.AddWithValue("@Отчество", textBox3.Text);

command.Parameters.AddWithValue("@Страна", textBox4.Text);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

disp\_Авторы\_Click(sender, e);

statText.Text = "Запись успешно добавлена. Используйте \"Создать транзакцию\", чтобы зафиксировать изменения";

if (dataGridView1.Rows.Count > 0)

{

int lastIndex = dataGridView1.Rows.Count - 1;

dataGridView1.FirstDisplayedScrollingRowIndex = lastIndex;

dataGridView1.Rows[lastIndex].Selected = true;

}

}

else

{

MessageBox.Show("Не удалось добавить запись.");

}

}

}

private void search\_author\_Click(object sender, EventArgs e)

{

doSearchRoutine("Авторы", "Фамилия", textBox1.Text);

}

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

{

doSearchRoutine("Авторы", "Страна автора", textBox4.Text);

}

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

{

DoTransactRoutine("Авторы");

}

private void disp\_Авторы\_Click(object sender, EventArgs e)

{

DoDisplayRoutine("Авторы");

}

//Издательства ========================

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

{

string que = "INSERT INTO Издательство VALUES (@Название, @Город, @Адрес)";

using (SqlCommand command = new SqlCommand(que, connection))

{

command.Parameters.AddWithValue("@Название", textBox5.Text);

command.Parameters.AddWithValue("@Город", textBox6.Text);

command.Parameters.AddWithValue("@Адрес", textBox7.Text);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

dispIzd\_Click(sender, e);

statText.Text = "Запись успешно добавлена. Используйте \"Создать транзакцию\", чтобы зафиксировать изменения";

if (dataGridView1.Rows.Count > 0)

{

int lastIndex = dataGridView1.Rows.Count - 1;

dataGridView1.FirstDisplayedScrollingRowIndex = lastIndex;

dataGridView1.Rows[lastIndex].Selected = true;

}

}

else

{

MessageBox.Show("Не удалось добавить запись.");

}

}

}

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

{

DoTransactRoutine("Издательство");

}

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

{

DoDisplayRoutine("Издательство");

}

//Жанры================================

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

{

string que = "INSERT INTO Жанр VALUES (@GENRE)";

using (SqlCommand command = new SqlCommand(que, connection))

{

command.Parameters.AddWithValue("@GENRE", textBox8.Text);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

dispGenre\_Click(sender, e);

statText.Text = "Запись успешно добавлена. Используйте \"Создать транзакцию\", чтобы зафиксировать изменения";

if (dataGridView1.Rows.Count > 0)

{

int lastIndex = dataGridView1.Rows.Count - 1;

dataGridView1.FirstDisplayedScrollingRowIndex = lastIndex;

dataGridView1.Rows[lastIndex].Selected = true;

}

}

else

{

MessageBox.Show("Не удалось добавить запись.");

}

}

}

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

{

doSearchRoutine("Жанр", "Название жанра", textBox8.Text);

}

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

{

DoTransactRoutine("Жанр");

}

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

{

DoDisplayRoutine("Жанр");

}

//КНИГА=============================

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

{

string que = "INSERT INTO Книга([Название],[Год выпуска],[Число страниц],[Язык книги],[Цена]) VALUES (@name,@yearIs,@pagesCol,@lang,@moneh)";

using (SqlCommand command = new SqlCommand(que, connection))

{

command.Parameters.AddWithValue("@name", textBox9.Text);

command.Parameters.AddWithValue("@yearIs", textBox10.Text);

command.Parameters.AddWithValue("@pagesCol", textBox11.Text);

command.Parameters.AddWithValue("@lang", textBox12.Text);

command.Parameters.AddWithValue("@moneh", textBox13.Text);

try

{

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

dispBook\_Click(sender, e);

statText.Text = "Запись успешно добавлена. Используйте \"Создать транзакцию\", чтобы зафиксировать изменения";

if (dataGridView1.Rows.Count > 0)

{

int lastIndex = dataGridView1.Rows.Count - 1;

dataGridView1.FirstDisplayedScrollingRowIndex = lastIndex;

dataGridView1.Rows[lastIndex].Selected = true;

}

}

}catch

{

MessageBox.Show("Не удалось добавить запись. Проверьте корректность введенных данных");

}

}

}

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

{

doSearchRoutine("Книга", "Название", textBox9.Text);

}

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

{

DoTransactRoutine("Книга");

}

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

{

DoDisplayRoutine("Книга");

dataGridView1.Columns[1].Visible = false;

dataGridView1.Columns[2].Visible = false;

dataGridView1.Columns[3].Visible = false;

}

}}

# ПРИЛОЖЕНИЕ В

## **Код C#, выполняющий логику регистрации и авторизации**

using System;

using System.IO;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Drawing;

using System.Threading;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class AuthForm : Form

{

private SqlConnection connection;

public SqlConnection GetConnection

{

get { return connection; }

}

private List<string> acessLevels = new List<string>();

public string[] GetAcessLevels {

get { return acessLevels.ToArray(); }

}

private string DateSourceBind ;

public void WhichWindow(string op)

{

this.AutoSize = true;

if (op == "register")

{

regbox.Visible = true;

regbox.Location = new Point(5, 5);

regbox.Size = new Size(213, 340);

}

else if (op == "authorize")

{

authbox.Visible = true;

authbox.Location = new Point(5, 5);

authbox.Size = new Size(213, 340);

}

}

public AuthForm()

{

InitializeComponent();

this.AutoSizeMode = AutoSizeMode.GrowAndShrink;

this.AutoSize = true;

DateSourceBind = Form1.GetDataSources;

}

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

{

try

{

string connectionString = $"Data Source={DateSourceBind};Initial Catalog=Библиотека;Integrated Security=True";

string transfer = textBox1.Text;

transfer = transfer.Replace(" ", "\_");

transfer += "\_" + comboBox1.Text;

string query = $"CREATE LOGIN {textBox2.Text} WITH PASSWORD = '{textBox5.Text}'; " +

$"CREATE USER {transfer} FOR LOGIN {textBox2.Text}; " +

$"GRANT SELECT ON Авторы TO {textBox2.Text};";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

connection.Open();

command.ExecuteNonQueryAsync();

connection.Close();

}

MessageBox.Show($"Пользователь {textBox2.Text} успешно зарегистрирован", "RegGud", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (SqlException er)

{

MessageBox.Show($"Невозможно добавить такого пользователя\n" + er, "RegErr", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

WhichWindow("authorize");

}

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

{

this.AutoSize = true;

AcceptButton = button1;

AcceptButton = button2;

comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;

}

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

{

string PerformConnectionString(string name, string code)

{

string cd = $"Data Source={DateSourceBind};Initial Catalog=Библиотека;User ID={name};Password='{code}';";

return cd;

}

try

{

string connectionString = PerformConnectionString(textBox3.Text, textBox4.Text);

connection = new SqlConnection(connectionString);

await connection.OpenAsync();

MessageBox.Show("Успешно подключено", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

using (SqlCommand command = new SqlCommand($"USE Библиотека SELECT DP2.name as DatabaseRoleName FROM sys.server\_principals AS SP JOIN sys.database\_principals AS DP ON SP.sid = DP.sid JOIN sys.database\_role\_members AS DRM ON DP.principal\_id = DRM.member\_principal\_id JOIN sys.database\_principals AS DP2 ON DRM.role\_principal\_id = DP2.principal\_id WHERE SP.name = '{textBox3.Text}' ORDER BY DP2.name;", connection))

{

using (SqlDataReader reader = await command.ExecuteReaderAsync())

{

if (reader.HasRows)

{

while (reader.Read())

{

string roleName = reader["DatabaseRoleName"].ToString();

acessLevels.Add(roleName);

}

}

}

Console.WriteLine();

}

DialogResult = DialogResult.OK;

this.Close();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка подключения: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

//DialogResult = DialogResult.Retry;

textBox3.Text = "";

textBox4.Text = "";

textBox3.Focus();

}

}

private void authbox\_Enter(object sender, EventArgs e)

{

}}}

# ПРИЛОЖЕНИЕ Г

## **Результат проверки на плагиат**

