МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гомельский государственный университет

имени Франциска Скорины»

Факультет математики и технологий программирования

Кафедра математических проблем управления и информатики

Допущена к защите

Зав. кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.С. Смородин

"\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г.

**Разработка приложения для автоматизации работы библиотеки**

Дипломная работа

Исполнитель

студент группы ПИ-41 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.М. Обухов

Научный руководитель

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

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4

1 Назначение разработанного приложения 4

2 Описание средств разработки приложения **Ошибка! Закладка не определена.**

2.1 Среда разработки Visual Studio 2022 4

2.2 Система Управления Базами Данных Microsoft SQL Server 4

2.3 Среда управления SQL Server Management Studio 4

2.4 Язык программирования SQL 4

2.5 Язык программирования C# 4

3 Структура базы данных разработанного приложения 4

3.1 Состав таблиц базы данных 4

3.2 Структура таблиц базы данных 4

4 Взаимодействие пользователя с разработанным приложением 4

ЗАКЛЮЧЕНИЕ 4

ПРИЛОЖЕНИЕ А Скрипт для создания структуры таблиц базы данных 4

ПРИЛОЖЕНИЕ Б Код C#, выполняющий подключение к СУБД, и выполняющий ряд заданных действий для взаимодействия с СУБД 4

ПРИЛОЖЕНИЕ В Код C#, выполняющий логику регистрации и авторизации 4

ПРИЛОЖЕНИЕ В Код C#, выполняющий регистрацию пользователя на стороне сервера 4

ПРИЛОЖЕНИЕ Д Результат проверки на плагиат 4

# ВВЕДЕНИЕ

В библиотеках хранится огромное множество 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), а также Android-приложения для просмотра каталога книг на языке Java (см. приложение Б).

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, Android Studio. Для хранения и получения данных будет подключена СУБД на MS SQL Server.

# Назначение и возможности разработанных приложений

Разработанные приложения, включая оконное приложение на языке C# и Android-приложение на языке Java, предоставляют пользователям широкий спектр возможностей для взаимодействия с данными библиотеки. Эти приложения предназначены для использования участниками библиотеки, такими как читатели и сотрудники, с целью эффективного ведения учета книг, управления поставками и списаниями, а также для быстрого доступа к информации о книгах и их выдаче.

Оконное приложение на языке C# предоставляет пользователям возможность быстрого и удобного поиска и просмотра информации о книгах. Обычные пользователи могут воспользоваться текстовым поиском для нахождения нужной книги или ознакомиться со списком книг, которые чаще всего берут читатели. Это приложение предлагает интуитивно понятный интерфейс, позволяющий легко ориентироваться в библиотечном каталоге и получать доступ к необходимой информации.

Для пользователей, имеющих учетные записи с расширенными правами доступа, предоставляются дополнительные возможности управления данными. В приложении предусмотрены специальные окна и инструменты для просмотра и управления содержимым базы данных. Эти пользователи могут выполнять следующие операции:

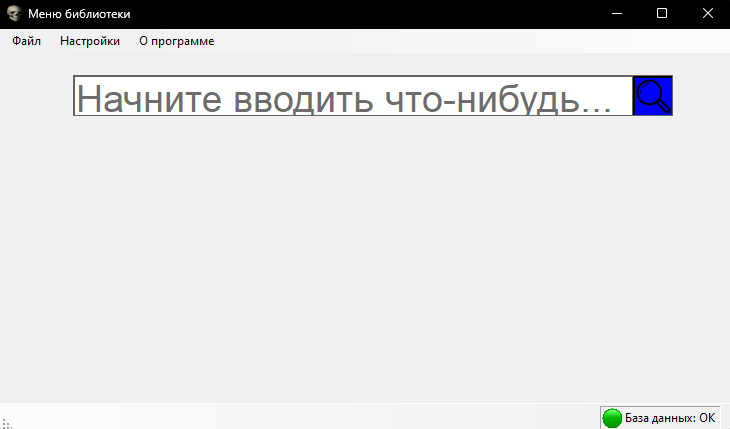
* Просмотр и редактирование личных данных читателей и сотрудников;
* Управление информацией о книгах, включая название, жанр, издательство, инвентарный номер и другие параметры;
* Ведение учета выдачи книг читателям, что позволяет контролировать ответственность читателей за полученные книги;
* Генерация и просмотр отчетов по использованию ресурсов библиотеки, что помогает анализировать активность пользователей и состояние библиотечного фонда.

Android-приложение, разработанное на языке Java, предназначено для пользователей, которые хотят иметь доступ к библиотечному каталогу через мобильные устройства. Это приложение предоставляет следующие возможности:

* Просмотр списка доступных книг в библиотеке;
* Поиск книг по различным критериям, таким как название, автор, жанр и другие параметры.

Интуитивно понятный интерфейс делает приложения удобными для пользователей разного уровня владения устройствами. Они обеспечивают быстрый доступ к необходимой информации о книгах, читателях и сотрудниках, что позволяет эффективно управлять библиотечным фондом. Гибкость и масштабируемость оконного приложения на C# позволяют пользователям с разными уровнями доступа выполнять широкий спектр задач, что делает его универсальным инструментом для управления библиотекой. Android-приложение предоставляет возможность доступа к библиотечному каталогу в любое время и в любом месте, что особенно удобно для пользователей, предпочитающих мобильные устройства.

Разработанные приложения обеспечивают комплексный подход к управлению библиотечными данными и предоставляют пользователям широкий спектр возможностей для взаимодействия с библиотечным фондом. Обычные пользователи могут легко находить и просматривать книги, а пользователи с расширенными правами доступа имеют инструменты для эффективного управления всеми аспектами библиотеки. Android-приложение дополняет возможности оконного приложения, предоставляя мобильный доступ к каталогу книг. Все эти функции направлены на повышение удобства и эффективности работы библиотеки, удовлетворяя потребности как читателей, так и сотрудников.



**Рисунок 1 – Рабочая область пользовательского интерфейса оконного приложения c# для поиска книг**



**Рисунок 2 – Рабочая область пользовательского интерфейса мобильного приложения для просмотра каталога книг**

# Средства разработки приложений

## **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.

## **Среда разработки Android Studio**

Android Studio - интегрированная среда разработки (IDE) для работы с платформой Android, анонсированная 16 мая 2013 года на конференции Google I/O. В последней версии Android Studio поддерживается Android 4.1 и выше.

Данная IDE находилась в свободном доступе начиная с версии 0.1, опубликованной в мае 2013, а затем перешла в стадию бета-тестирования, начиная с версии 0.8, которая была выпущена в июне 2014 года. Первая стабильная версия 1.0 была выпущена в декабре 2014 года, тогда же прекратилась поддержка плагина Android Development Tools (ADT) для Eclipse.

Android Studio, основанная на программном обеспечении IntelliJ IDEA от компании JetBrains, - официальное средство разработки Android приложений[1]. Данная среда разработки доступна для Android,Windows, macOS и GNU/Linux[2]. 17 мая 2017, на ежегодной конференции Google I/O, Google анонсировал поддержку языка Kotlin, используемого в Android Studio, как официального языка программирования для платформы Android в дополнение к Java и C++.

Новые функции появляются с каждой новой версией Android Studio. На данный момент доступны следующие функции:

* Расширенный редактор макетов: WYSIWYG, способность работать с UI-компонентами при помощи Drag-and-Drop, функция предпросмотра макета на нескольких конфигурациях экрана.
* Сборка приложений, основанная на Gradle.
* Различные виды сборок и генерация нескольких .apk-файлов.
* Рефакторинг кода
* Статический анализатор кода (Lint), позволяющий находить проблемы производительности, несовместимости версий и другое.
* Встроенный ProGuard и утилита для подписывания приложений.
* Шаблоны основных макетов и компонентов Android.
* Поддержка разработки приложений для Android Wear и Android TV.
* Встроенная поддержка Google Cloud Platform, которая включает в себя интеграцию с сервисами Google Cloud Messaging и App Engine.
* Android Studio 2.1 поддерживает Android N Preview SDK, а это значит, что разработчики смогут начать работу по созданию приложения для новой программной платформы.
* Новая версия Android Studio 2.1 способна работать с обновленным компилятором Jack, а также получила улучшенную поддержку Java 8 и усовершенствованную функцию Instant Run.
* Начиная с Platform-tools 23.1.0 для Linux - исключительно 64-разрядная.
* В Android Studio 3.0 по стандарту включены инструменты языка Kotlin, основанные на JetBrains IDE.

## **Язык программирования Java**

Java — язык программирования общего назначения. Он используется для написания кода, который может выполняться на разных платформах: компьютерах, мобильных устройствах и серверах.

По результатам рейтинга State of Octoverse 2022 Java входит в тройку самых популярных языков программирования. Java опережает по популярности PHP, C#, C++, TypeScript и другие востребованные языки, а уступает только JavaScript и Python. В индексе TIOBE на июль 2023 года Java занимает четвертое место.

Java разработала компания Sun Microsystems в начале 90-х годов XX века. Ведущую роль в создании языка сыграл канадский инженер Джеймс Гослинг, а на ранних этапах разработки язык назывался Oak. Затем его переименовали в честь сорта кофе Java и в честь этого оставили чашку в логотипе.

Создатели планировали использовать Java для программирования бытовой электроники, однако практически сразу после выпуска версии 1.0 в 1995 году язык стали использовать разработчики серверного и клиентского ПО. В 2010 году компанию Sun Microsystems купила Oracle.

В Java много внимания уделено раннему обнаружению ошибок и динамической проверке во время работы программы. Поэтому язык считается безопасным и на нем часто пишут важные системы: банковские терминалы, системы обработки транзакций, сервисы координации перелетов и другие.

Кроме того, Java достаточно дешевый в обслуживании — запускать код и работать с ним можно практически с любого компьютера, вне зависимости от конкретной аппаратной инфраструктуры. В том числе поэтому язык популярен в промышленной разработке, то есть в крупных компаниях.

Чаще всего язык программирования используется для создания серверных приложений разной степени сложности и направленности: это могут быть как отдельные приложения, так и вся серверная часть проекта. Также на Java пишут программы для финансовых организаций, которые обеспечивают проведение транзакций, фиксацию торговых операций.

# Структура базы данных приложения

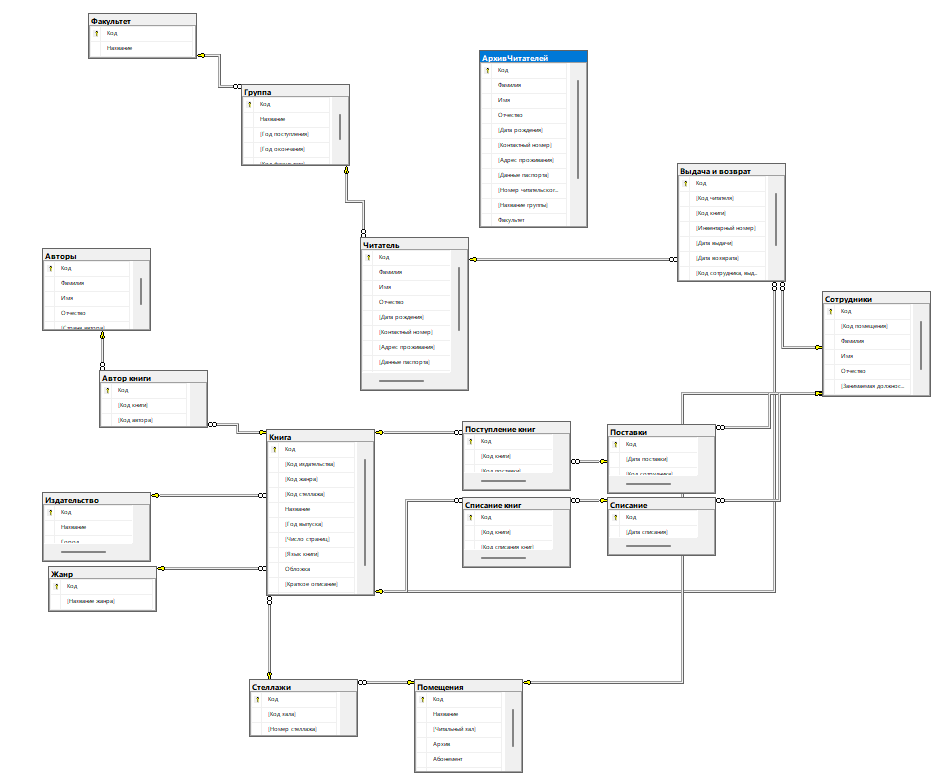
## **Состав таблиц базы данных**

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 и возврат | 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 |
| 1. Группа | Информация о группах |
| 1. Факультет | Информация о факультетах |
| 1. АрхивЧитателей | Данные о студентах, которые больше не учатся в университете, и их задолженностях |

Взаимосвязь между таблицами указана на рисунке 3.



**Рисунок 3 – Схема базы данных**

## **Структура таблиц базы данных**

В таблицах 2-17 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 |
| Обложка | varbinary(max) | Изображение обложки книги |
| Краткое описание | varchar(4000) | Краткое описание книги |
| 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 и возврат**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Уникальный код выдачи |
| Код читателя | int | Код читателя |
| Код книги | int | Код книги |
| 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, выдавшего книгу |
| Книга утеряна | bit | Отметка об утере 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 |
| 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 |
| Код группы | int | Код группы |
| Имя для входа | varchar(100) | Имя учетной записи |

**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 |

**Окончание таблицы 15**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Занимаемая должность | varchar(40) | Занимаемая сотрудником должность |
| Стаж | int | Стаж сотрудника (в годах) |

**Таблица 16 – Группа**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Уникальный код группы |
| Название | varchar(40) | Название группы |
| Год поступления | int | Год поступления студентов группы |
| Год окончания | int | Год выпуска студентов группы |
| Код факультета | varchar(40) | Код факультета группы |

**Таблица 17 – Факультеты**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Уникальный код факультета |
| Название | varchar(40) | Название факультета |

**Таблица 18 – АрхивЧитателей**

|  |  |  |
| --- | --- | --- |
| Наименование поля | Формат поля | Содержимое поля |
| Код | int | Код записи |
| Фамилия | varchar(40) | Фамилия студента |
| Имя | varchar(40) | Имя студента |
| Отчество | varchar(40) | Отчество студента |
| Дата рождения | date | Дата рождения студента |
| Контактный номер | varchar(40) | Номер для связи со студентом |
| Адрес проживания | varchar(40) | Адрес проживания студента |
| Данные паспорта | varchar(60) | Данные паспорта студента |
| Номер читательского билета | varchar(40) | Номер читательского билета студента |
| Название группы | varchar(40) | Название группы, где обучался студент |
| Факультет | varchar(40) | Название факультета, где обучался студент |

**Окончание таблицы 18**

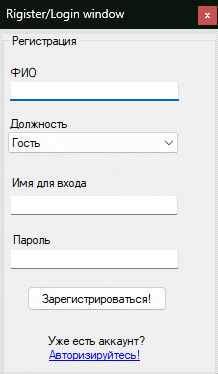
|  |  |  |
| --- | --- | --- |
| Задолженности | varchar(500) | Книги, которые не вернул студент на момент выпуска |
| Имя для входа | varchar(100) | Имя для входа, под которым был зарегистрирован студент |

# Взаимодействие пользователя с разработанным приложением

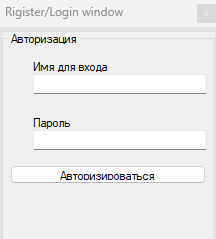
# 4.1 Оконное приложение Windows Forms, язык программирования C#

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

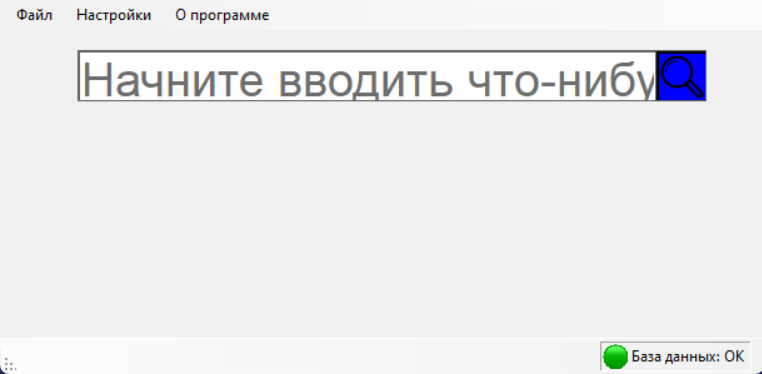
Предварительно, пользователь может зарегистрироваться или авторизироваться, чтобы получить доступ к дополнительным возможностям приложения. (По умолчанию активирован базовый функционал: только просмотр книг, авторов, издательств – публичная информация)



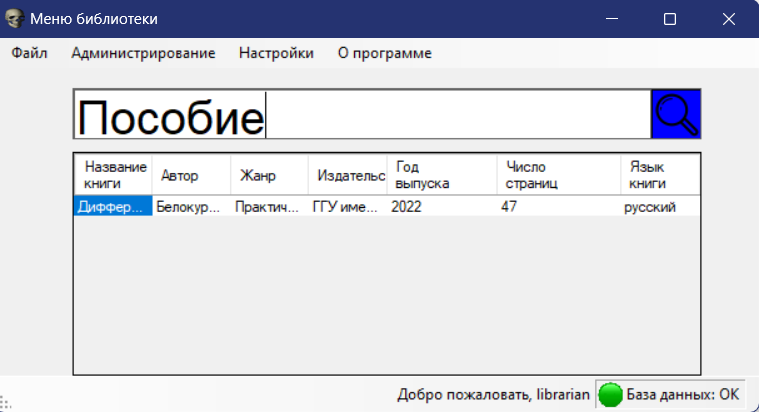
**Рисунок 4 – Регистрационная форма**



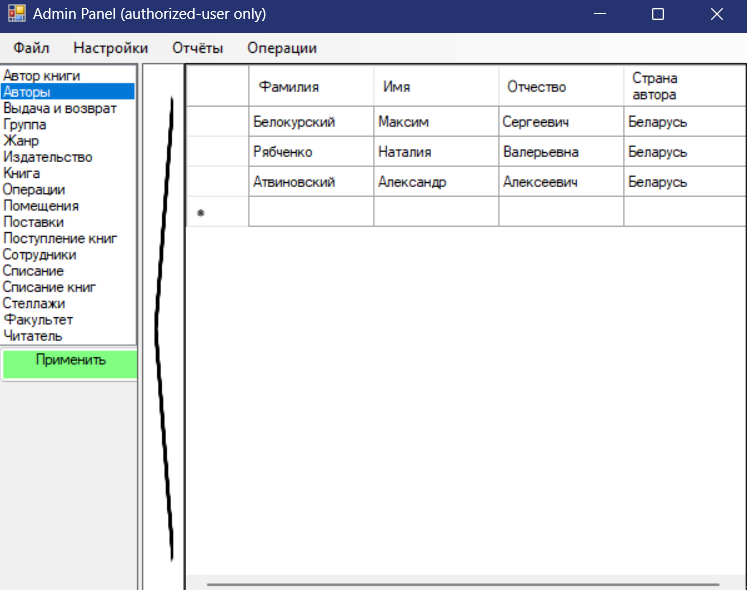
**Рисунок 5 – Форма авторизации пользователя**

****

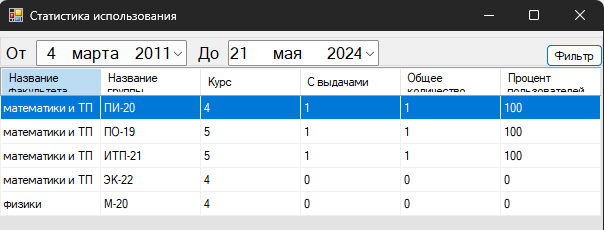
**Рисунок 6 – Рабочая область пользовательского интерфейса для взаимодействия с пользователем**

****

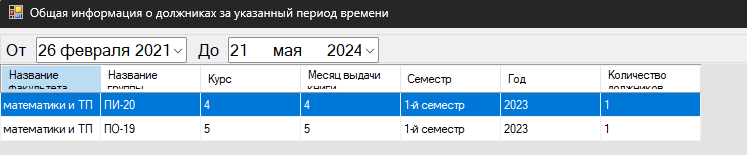
**Рисунок 7 – Рабочая область пользовательского интерфейса для взаимодействия с пользователем (пользователь вошёл в систему, есть права администратора)**

****

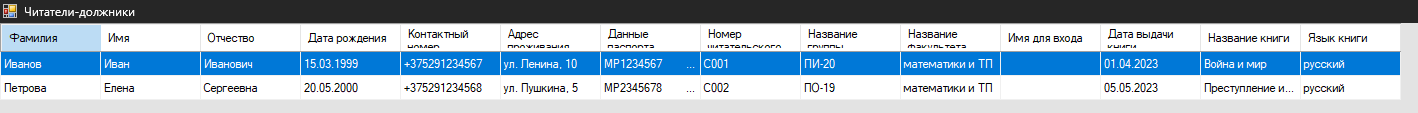
**Рисунок 8 – Рабочая область пользовательского интерфейса для управления данными в базе данных (пользователь вошёл в систему, и у него есть права управлять базой данных)**

****

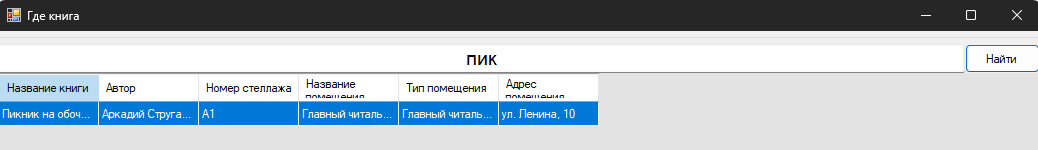
**Рисунок 9 – Отчет о пользовании библиотекой группами по факультетам за выбранный период времени**

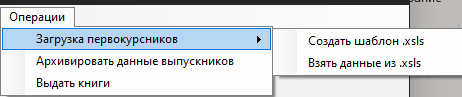
****

**Рисунок 10 – Общая информация по должникам за указанный период времени.**

****

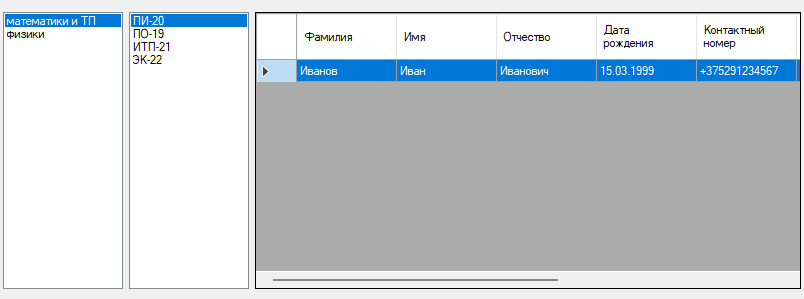
**Рисунок 11 – Отчет о читателях-должниках**

**Рисунок 11 – Форма для установления местонахождения указанных книг**

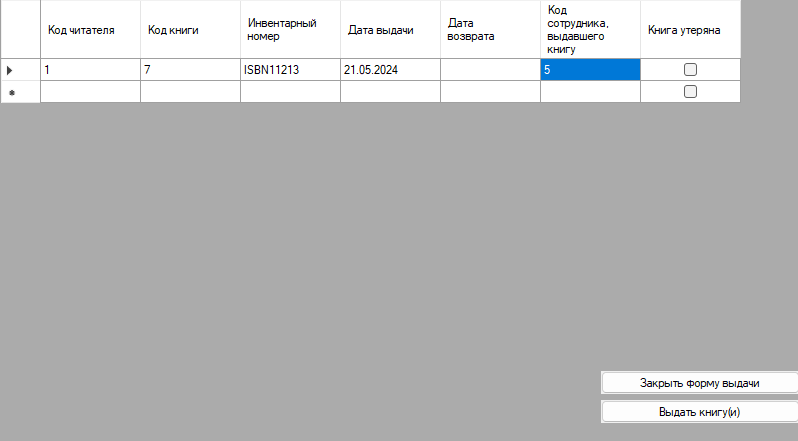


**Рисунок 12 – Список доступных операций.**

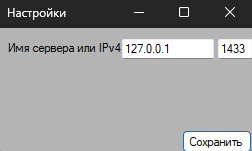
Загрузка первокурсников позволяет создать шаблон Excel для загрузки новых студентов-первокурсников в базу данных, и загрузить из в базу данных.  
 Архивация данных выпускников выполняет перенос данных выпускников с их задолженностями в отдельную таблицу. Это необходимо, чтобы потребовать от студента вернуть все книги, взятые им в библиотеке, в библиотечный фонд.



**Рисунок 13 – Форма выдачи книг. Выбор студента, которому будут выдавать книги**

****

**Рисунок 13 – Форма выдачи книг. Выбор книг, которые будут выданы студенту, указание даты выдачи и ответственного за выдачу сотрудника**

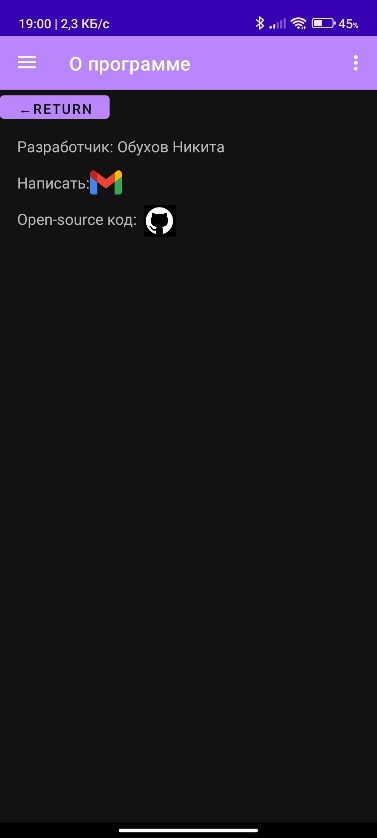
****

**Рисунок 13 – Форма настроек. Настройки сохраняются даже после выхода из приложения**

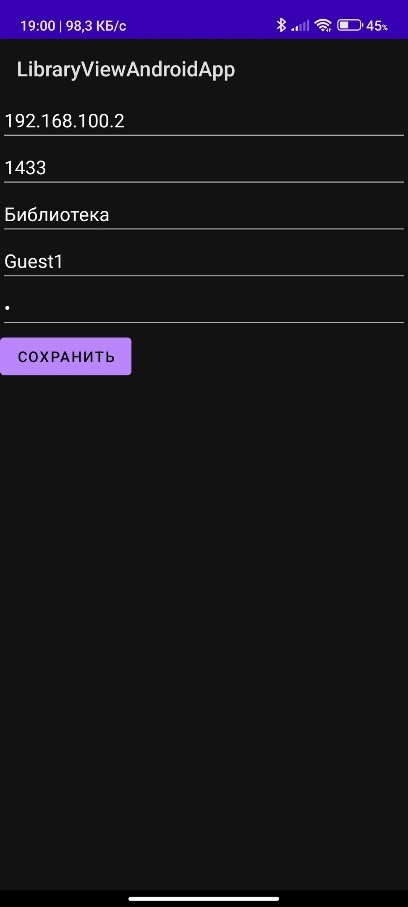
# 4.2 Мобильное приложение Android на языке Java



**Рисунок 14 – Рабочая область пользовательского интерфейса мобильного приложения для просмотра каталога книг**



**Рисунок 15 – Окно «О программе»**



**Рисунок 16 – Окно настроек подключения**

# 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.

Разработанные приложения обеспечивают комплексный подход к управлению библиотечными данными и предоставляют пользователям широкий спектр возможностей для взаимодействия с библиотечным фондом. Обычные пользователи могут легко находить и просматривать книги, а пользователи с расширенными правами доступа имеют инструменты для эффективного управления всеми аспектами библиотеки. Android-приложение дополняет возможности оконного приложения, предоставляя мобильный доступ к каталогу книг. Все эти функции направлены на повышение удобства и эффективности работы библиотеки, удовлетворяя потребности как читателей, так и сотрудников.

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

use Библиотека;

DROP PROCEDURE GetUserRoles;

DROP PROCEDURE GetUserTablePermissions;

DROP PROCEDURE RemoveOutdatedReaders;

DROP PROCEDURE GetPercentageUsersByGroupAndFaculty;

DROP PROCEDURE GetDebtorsReport

DROP PROCEDURE GetReaderInfoExact

DROP FUNCTION BooksPublicInfo

DROP PROCEDURE dbo.GetDependentTableName;

DROP PROCEDURE FindBookLocation;

DROP TABLE АрхивЧитателей

alter table Книга

drop constraint FK\_Книга\_Издательство,

constraint FK\_Книга\_Жанр,

constraint FK\_Книга\_Стеллаж

alter table [Автор книги]

drop constraint FK\_Книга\_Автор\_книги,

constraint FK\_Автор\_книги\_Авторы

alter table [Поступление книг]

drop constraint FK\_Книга\_Поступления\_книг,

constraint FK\_Поступление\_книг\_Поставки

alter table [Списание книг]

drop constraint FK\_Книга\_Списание\_книг,

constraint FK\_Списание\_книг\_Списание

alter table [Выдача и возврат]

drop constraint FK\_Выдача\_и\_возврат\_Книга,

constraint FK\_Выдача\_и\_возврат\_Сотрудники,

constraint FK\_Выдача\_и\_возврат\_Читатель

alter table [Стеллажи]

drop constraint FK\_Стеллажи\_Помещения

alter table [Сотрудники]

drop constraint FK\_Сотрудники\_Помещения

alter table [Поставки]

drop constraint FK\_Поставки\_Сотрудники

alter table [Списание]

drop constraint FK\_Списание\_Сотрудники

alter table Группа

drop constraint FK\_Группа\_Факультеты

alter table Читатель

drop constraint FK\_Читатель\_Группы

drop table Издательство

drop table Жанр

drop table Книга

drop table [Автор книги]

drop table Авторы

drop table [Поступление книг]

drop table Поставки

drop table [Списание книг]

drop table Списание

drop table [Выдача и возврат]

drop table Стеллажи

drop table Читатель

drop table [Помещения]

drop table Сотрудники

drop table Факультет

drop table Группа

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,

[Обложка] VARBINARY(MAX),

[Краткое описание] varchar(4000),

[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 и возврат]

(

Код int primary key not null identity,

[Код читателя] int,

[Код книги] int,

[Инвентарный номер] varchar(40),

[Дата выдачи] date not null,

[Дата возврата] date,

[Код сотрудника, выдавшего книгу] int,

[Книга утеряна] bit

)

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),

[Код группы] INT,

[Имя для входа] VARCHAR(100)

);

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

CREATE TABLE Группа

(

Код INT PRIMARY KEY NOT NULL IDENTITY,

Название VARCHAR(40) NOT NULL,

[Год поступления] INT NOT NULL,

[Год окончания] INT NOT NULL,

[Код факультета] INT NOT NULL

);

CREATE TABLE Факультет

(

Код INT PRIMARY KEY NOT NULL IDENTITY,

Название VARCHAR(40) NOT NULL

); go

CREATE TABLE АрхивЧитателей (

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),

[Название группы] varchar(20),

[Факультет] varchar(20),

[Задолженности] varchar(500),

[Имя для входа] VARCHAR(100)

);

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 Поставки(Код)

GO

alter table [Списание книг]

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

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

GO

alter table [Выдача и возврат]

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

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

constraint FK\_Выдача\_и\_возврат\_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

alter table Читатель

add constraint FK\_Читатель\_Группы foreign key ([Код группы]) references Группа(Код)

GO

alter table Группа

add constraint FK\_Группа\_Факультеты foreign key ([Код факультета]) references Факультет(Код)

GO

-- Для таблицы Жанр

INSERT INTO Жанр ([Название жанра])

VALUES

('Фантастика'),

('Роман'),

('Детектив'),

('Поэзия');

-- Для таблицы Авторы

INSERT INTO Авторы ([Фамилия], [Имя], [Отчество], [Страна автора])

VALUES

('Толстой', 'Лев', 'Николаевич', 'Россия'),

('Достоевский', 'Федор', 'Михайлович', 'Россия'),

('Пушкин', 'Александр', 'Сергеевич', 'Россия'),

('Чехов', 'Антон', 'Павлович', 'Россия'),

('Аркадий', 'Стругацкий', '', 'Россия'),

('Борис', 'Стругацкий', '', 'Россия');

-- Для таблицы Помещения

INSERT INTO Помещения ([Название], [Адрес помещения])

VALUES

('Главный читальный зал', 'ул. Ленина, 10'),

('Детский отдел', 'ул. Пушкина, 5'),

('Абонемент', 'ул. Кирова, 15');

-- Для таблицы Стеллажи

INSERT INTO Стеллажи ([Код зала], [Номер стеллажа])

VALUES

(1, 'A1'),

(1, 'A2'),

(1, 'B1'),

(2, 'C1'),

(3, 'A1');

INSERT INTO Издательство values

('АСТ', 'Москва', 'Междугородняя 6'),

('ГГУ им. Ф. Скорины', 'Гомель', 'Советская ?')

-- Для таблицы Книга

INSERT INTO Книга ([Код издательства], [Код жанра], [Код стеллажа], [Название], [Год выпуска], [Число страниц], [Язык книги], [Обложка], [Краткое описание], [Цена])

VALUES

(1, 1, 1, 'Война и мир', 1869, 1225, 'русский', NULL, 'Эпический роман Льва Толстого о войне 1812 года', 25.99),

(1, 1, 1, 'Война и мир', 1869, 1225, 'русский', NULL, 'Эпический роман Льва Толстого о войне 1812 года', 25.99),

(2, 2, 2, 'Преступление и наказание', 1866, 551, 'русский', NULL, 'Роман Федора Достоевского о преступлении и наказании 987897897987897854848546548756 714273173213213 213212312313213213213213213213213213215649674987987', 19.99),

(2, 3, 3, 'Евгений Онегин', 1833, 368, 'русский', NULL, 'Роман Александра Пушкина', 14.99),

(1, 4, 4, 'Дама с собачкой', 1899, 192, 'русский', null, 'Повесть Антона Чехова', 12.99),

(1,3,1,'Пикник на обочине', 2008, 48, 'русский', (SELECT BulkColumn FROM Openrowset(Bulk 'G:\content.jpg', Single\_Blob) as Image),'легендарная философско-фантастическая книга',30),

(1,3,1,'Пикник на обочине', 2008, 48, 'русский', (SELECT BulkColumn FROM Openrowset(Bulk 'G:\content.jpg', Single\_Blob) as Image),'легендарная философско-фантастическая книга',30),

(1,3,1,'Пикник на обочине', 2008, 48, 'русский', (SELECT BulkColumn FROM Openrowset(Bulk 'G:\content.jpg', Single\_Blob) as Image),'легендарная философско-фантастическая книга',30)

-- (SELECT BulkColumn FROM Openrowset(Bulk 'D:\Downloads\43w8XwCrfeQ.jpg', Single\_Blob) as Image)

-- Для таблицы [Автор книги]

INSERT INTO [Автор книги] ([Код книги], [Код автора])

VALUES

(1, 1),

(2, 1),

(3, 2),

(4, 3),

(5, 4),

(6,5),

(6,6),

(7,5),

(7,6),

(8,5),

(8,6)

-- Для таблицы Сотрудники

INSERT INTO Сотрудники ([Код помещения], [Фамилия], [Имя], [Отчество], [Занимаемая должность], [Стаж])

VALUES

(1, 'Иванов', 'Петр', 'Ильич', 'Библиотекарь', 5),

(1, 'Петров', 'Иван', 'Сергеевич', 'Администратор', 3),

(2, 'Сидорова', 'Анна', 'Петровна', 'Библиотекарь', 4),

(3, 'Козлова', 'Елена', 'Ивановна', 'Кассир', 2);

-- Для таблицы Поставки

INSERT INTO Поставки ([Дата поставки], [Код сотрудника], [Поставщик])

VALUES

('2023-01-15', 1, 'Книгоиздательство "Русский Классик"'),

('2023-02-20', 2, 'Издательский дом "Москва"'),

('2023-03-25', 3, 'Издательство "Огонек"');

-- Для таблицы [Поступление книг]

INSERT INTO [Поступление книг] ([Код книги], [Код поставки], [Число книг])

VALUES

(1, 1, 20),

(2, 2, 15),

(3, 3, 30),

(4, 1, 25);

-- Для таблицы Факультет

INSERT INTO Факультет (Название)

VALUES

('математики и ТП'),

('физики');

-- Для таблицы Группа

INSERT INTO Группа (Название, [Год поступления], [Год окончания], [Код факультета])

VALUES

('ПИ-20', 2020, 2023, 1),

('ПО-19', 2019, 2023, 1),

('ИТП-21', 2019, 2025, 1),

('ЭК-22', 2020, 2025, 1),

('М-20', 2020, 2024, 2);

-- Для таблицы Читатель

INSERT INTO Читатель ([Фамилия], [Имя], [Отчество], [Дата рождения], [Контактный номер], [Адрес проживания], [Данные паспорта], [Номер читательского билета], [Код группы])

VALUES

('Иванов', 'Иван', 'Иванович', '1999-03-15', '+375291234567', 'ул. Ленина, 10', 'MP1234567', 'C001', 1),

('Петрова', 'Елена', 'Сергеевна', '2000-05-20', '+375291234568', 'ул. Пушкина, 5', 'MP2345678', 'C002', 2),

('Сидоров', 'Алексей', 'Петрович', '1998-07-25', '+375291234569', 'ул. Кирова, 15', 'MP3456789', 'C003', 3);

-- Для таблицы [Выдача и возврат]

INSERT INTO [Выдача и возврат] ([Код читателя], [Код книги], [Инвентарный номер], [Дата выдачи],[Дата возврата], [Код сотрудника, выдавшего книгу])

VALUES

(1, 1, 'INV001', '2023-04-01',null, 1),

(2, 3, 'INV002', '2023-05-05',null, 2),

(3, 4, 'INV003', '2023-06-10','2024-05-05', 3)

GO

--ПУБЛИЧНАЯ ИНФОРМАЦИЯ

CREATE FUNCTION BooksPublicInfo(

@SearchTerm VARCHAR(100)

)

RETURNS TABLE

AS

RETURN

(

SELECT

Книга.[Название],

Автор.Автор AS Автор,

Жанр.[Название жанра] AS Жанр,

Издательство.[Название] AS Издательство,

Книга.[Год выпуска],

Книга.[Число страниц],

Книга.[Язык книги],

Книга.[Обложка],

COUNT(DISTINCT CASE WHEN ([Выдача и возврат].[Дата выдачи] IS NOT NULL

AND [Выдача и возврат].[Дата возврата] IS NOT NULL

AND [Выдача и возврат].[Книга утеряна] IS NULL)

OR [Выдача и возврат].[Дата выдачи] IS NULL

THEN Книга.Код ELSE NULL END) AS Доступность,

COUNT(CASE WHEN [Выдача и возврат].[Дата выдачи] IS NOT NULL THEN [Выдача и возврат].[Код читателя] ELSE NULL END) AS [Как часто брали],

Книга.[Краткое описание]

FROM

Книга

JOIN

Жанр ON Книга.[Код жанра] = Жанр.Код

JOIN

[Издательство] ON Книга.[Код издательства] = [Издательство].Код

LEFT JOIN

(SELECT

[Автор книги].[Код книги],

STRING\_AGG([Авторы].[Фамилия] + ' ' + [Авторы].[Имя] + ' ' + Авторы.Отчество, ', ') AS Автор

FROM

[Автор книги]

JOIN

[Авторы] ON [Авторы].Код = [Автор книги].[Код автора]

GROUP BY

[Автор книги].[Код книги]

) AS Автор ON Автор.[Код книги] = Книга.Код

LEFT JOIN

[Выдача и возврат] ON [Выдача и возврат].[Код книги] = Книга.Код

WHERE

Книга.Название LIKE '%' + @SearchTerm + '%'

OR Автор.Автор LIKE '%' + @SearchTerm + '%'

OR Жанр.[Название жанра] LIKE '%' + @SearchTerm + '%'

GROUP BY

Книга.[Название],

Автор.Автор,

Жанр.[Название жанра],

Издательство.[Название],

Книга.[Год выпуска],

Книга.[Число страниц],

Книга.[Язык книги],

Книга.[Обложка],

Книга.[Краткое описание]

);

go

--select \* from BooksPublicInfo('');

--КОНЕЦ ПУБЛИЧНАЯ ИНФОРМАЦИЯ

GO

--СЛУЖЕБНЫЕ ОПЕРАЦИИ

IF EXISTS (SELECT 1 FROM sys.database\_principals WHERE name = 'Guest1')

BEGIN

DROP USER Guest1;

END

GO

IF EXISTS (SELECT 1 FROM sys.server\_principals WHERE name = 'Guest1')

BEGIN

DROP LOGIN Guest1;

END

GO

CREATE LOGIN Guest1 WITH PASSWORD = '1';

CREATE USER Guest1 FOR LOGIN Guest1;

GRANT SELECT ON Библиотека.dbo.BooksPublicInfo TO Guest1;

GO

--КОНЕЦ СЛУЖЕБНЫЕ ОПЕРАЦИИ

--СЛУЖЕБНАЯ ИНФОРМАЦИЯ

CREATE PROCEDURE GetUserTablePermissions

@DatabaseUserName NVARCHAR(255),

@TableName NVARCHAR(255)

AS

BEGIN

SELECT dp.permission\_name

FROM sys.database\_permissions dp

JOIN sys.database\_principals dpn ON dp.grantee\_principal\_id = dpn.principal\_id

WHERE dpn.name = (SELECT dp.name AS DatabaseUserName

FROM sys.database\_principals dp

JOIN sys.server\_principals sp ON dp.sid = sp.sid

WHERE sp.name = @DatabaseUserName) AND dp.major\_id = OBJECT\_ID(@TableName)

END;

go

CREATE PROCEDURE GetUserRoles

@LoginName NVARCHAR(100)

AS

BEGIN

DECLARE @UserId INT;

SELECT @UserId = dp.principal\_id

FROM sys.database\_principals dp

JOIN sys.server\_principals sp ON dp.sid = sp.sid

WHERE sp.name = @LoginName;

SELECT role.name AS RoleName

FROM sys.database\_role\_members members

JOIN sys.database\_principals role ON members.role\_principal\_id = role.principal\_id

WHERE members.member\_principal\_id = @UserId;

END;

go

CREATE PROCEDURE GetDependentTableName

@ParentTableName varchar(50),

@ColumnName varchar(50)

as

BEGIN

SELECT

OBJECT\_NAME(fk.referenced\_object\_id) AS ReferencedTable

FROM

sys.foreign\_key\_columns fkc

JOIN sys.foreign\_keys fk ON fkc.constraint\_object\_id = fk.object\_id

JOIN sys.columns c ON fkc.parent\_column\_id = c.column\_id AND fkc.parent\_object\_id = c.object\_id

WHERE

OBJECT\_NAME(fk.parent\_object\_id) = @ParentTableName

AND c.name = @ColumnName

END;

/\*

SELECT

fk.name AS ForeignKey,

OBJECT\_NAME(fk.parent\_object\_id) AS ParentTable,

c1.name AS ParentColumn,

OBJECT\_NAME(fk.referenced\_object\_id) AS ReferencedTable,

c2.name AS ReferencedColumn

FROM

sys.foreign\_keys AS fk

INNER JOIN

sys.foreign\_key\_columns AS fkc ON fk.OBJECT\_ID = fkc.constraint\_object\_id

INNER JOIN

sys.columns AS c1 ON fkc.parent\_object\_id = c1.object\_id AND fkc.parent\_column\_id = c1.column\_id

INNER JOIN

sys.columns AS c2 ON fkc.referenced\_object\_id = c2.object\_id AND fkc.referenced\_column\_id = c2.column\_id

WHERE

OBJECT\_NAME(fk.parent\_object\_id) = 'Книга'

\*/

--КОНЕЦ СЛУЖЕБНАЯ ИНФОРМАЦИЯ

go

--ОТЧЕТЫ

CREATE PROCEDURE dbo.GetPercentageUsersByGroupAndFaculty

(

@StartDate DATE,

@EndDate DATE

)

AS

BEGIN

WITH ВыдачиКниг AS (

SELECT

Ч.Код AS [Код читателя],

Г.[Код] AS [Код группы]

FROM

[Выдача и возврат] В

INNER JOIN Читатель Ч ON В.[Код читателя] = Ч.Код

INNER JOIN Группа Г ON Ч.[Код группы] = Г.[Код]

WHERE

В.[Дата выдачи] BETWEEN @StartDate AND @EndDate

),

ЧитателиСВыдачами AS (

SELECT

Г.[Код факультета],

Г.[Код] AS [Код группы],

COUNT(DISTINCT Ч.[Код]) AS С\_выдачами

FROM

Читатель Ч

INNER JOIN ВыдачиКниг ВК ON Ч.Код = ВК.[Код читателя]

INNER JOIN Группа Г ON Ч.[Код группы] = Г.[Код]

WHERE

Ч.[Код группы] IS NOT NULL

GROUP BY

Г.[Код факультета],

Г.[Код]

)

SELECT

Ф.[Название] AS [Название факультета],

Г.[Название] AS [Название группы],

CAST(YEAR(GETDATE()) - Г.[Год поступления] AS INT) AS [Курс],

ISNULL(ЧитателиСВыдачами.С\_выдачами, 0) AS [С выдачами],

ISNULL(Читатели.Общее\_количество, 0) AS [Общее количество],

IIF(ISNULL(Читатели.Общее\_количество, 0) = 0, 0, 100 \* CAST(ISNULL(ЧитателиСВыдачами.С\_выдачами, 0) AS FLOAT) / CAST(ISNULL(Читатели.Общее\_количество, 0) AS FLOAT)) AS [Процент пользователей от читателей]

FROM

Факультет Ф

INNER JOIN Группа Г ON Ф.[Код] = Г.[Код факультета]

LEFT JOIN ЧитателиСВыдачами ON Г.[Код] = ЧитателиСВыдачами.[Код группы] AND Ф.[Код] = ЧитателиСВыдачами.[Код факультета]

LEFT JOIN (

SELECT

Г.[Код факультета],

Г.[Код] AS [Код группы],

COUNT(DISTINCT Ч.[Код]) AS Общее\_количество

FROM

Читатель Ч

LEFT JOIN Группа Г ON Ч.[Код группы] = Г.[Код]

WHERE

Ч.[Код группы] IS NOT NULL

GROUP BY

Г.[Код факультета],

Г.[Код]

) AS Читатели ON Г.[Код] = Читатели.[Код группы] AND Ф.[Код] = Читатели.[Код факультета];

END;

go

CREATE PROCEDURE dbo.GetDebtorsReport

(

@StartDate DATETIME2,

@EndDate DATETIME2

)

AS

BEGIN

WITH ВыдачиКниг AS (

SELECT

Г.[Код факультета],

Ч.[Код группы],

MONTH(В.[Дата выдачи]) AS [Месяц],

CASE

WHEN MONTH(В.[Дата выдачи]) BETWEEN 1 AND 6 THEN '1-й семестр'

WHEN MONTH(В.[Дата выдачи]) BETWEEN 7 AND 12 THEN '2-й семестр'

END AS [Семестр],

YEAR(В.[Дата выдачи]) AS [Год],

COUNT(\*) AS [Количество должников]

FROM

Читатель Ч

INNER JOIN [Выдача и возврат] В ON Ч.[Код] = В.[Код читателя] AND В.[Дата возврата] IS NULL -- Должники

INNER JOIN Группа Г ON Ч.[Код группы] = Г.[Код]

WHERE

В.[Дата выдачи] BETWEEN @StartDate AND @EndDate

GROUP BY

Г.[Код факультета],

Ч.[Код группы],

MONTH(В.[Дата выдачи]),

CASE

WHEN MONTH(В.[Дата выдачи]) BETWEEN 1 AND 6 THEN '1-й семестр'

WHEN MONTH(В.[Дата выдачи]) BETWEEN 7 AND 12 THEN '2-й семестр'

END,

YEAR(В.[Дата выдачи])

)

SELECT

Ф.[Название] AS [Название факультета],

Г.[Название] AS [Название группы],

CAST(YEAR(GETDATE()) - Г.[Год поступления] AS INT) AS Курс,

VK.[Месяц] as [Месяц выдачи книги],

VK.[Семестр],

VK.[Год],

VK.[Количество должников]

FROM

ВыдачиКниг VK

INNER JOIN Группа Г ON VK.[Код группы] = Г.[Код]

INNER JOIN Факультет Ф ON VK.[Код факультета] = Ф.[Код];

END;

go

CREATE PROCEDURE dbo.GetReaderInfoExact

AS

BEGIN

SELECT

[Читатель].[Фамилия],

[Читатель].[Имя],

[Читатель].[Отчество],

[Читатель].[Дата рождения],

[Читатель].[Контактный номер],

[Читатель].[Адрес проживания],

[Читатель].[Данные паспорта],

[Читатель].[Номер читательского билета],

[Группа].[Название] as 'Название группы',

[Факультет].[Название] as 'Название факультета',

[Читатель].[Имя для входа],

[Выдача и возврат].[Дата выдачи] AS [Дата выдачи книги],

[Книга].[Название] AS [Название книги],

[Книга].[Язык книги] AS [Язык книги]

FROM [Читатель]

INNER JOIN [Выдача и возврат] ON [Читатель].[Код] = [Выдача и возврат].[Код читателя]

INNER JOIN [Книга] ON [Выдача и возврат].[Код книги] = [Книга].[Код]

INNER JOIN Группа ON Группа.Код = Читатель.[Код группы]

INNER JOIN Факультет ON Факультет.Код = Группа.[Код факультета]

WHERE [Выдача и возврат].[Дата возврата] IS NULL;

END;

GO

CREATE PROCEDURE FindBookLocation

@sterm NVARCHAR(100)

AS

BEGIN

SELECT

Книга.Название AS 'Название книги',

Авторы.Автор,

Стеллажи.[Номер стеллажа] AS 'Номер стеллажа',

Помещения.Название AS 'Название помещения',

Помещения.Название AS 'Тип помещения',

Помещения.[Адрес помещения] AS 'Адрес помещения'

FROM

Книга

INNER JOIN

(SELECT [Автор книги].[Код книги], STRING\_AGG([Авторы].[Фамилия] + ' ' + [Авторы].[Имя] + ' ' + Авторы.Отчество, ', ') AS Автор

FROM [Автор книги]

INNER JOIN [Авторы] ON [Автор книги].[Код автора] = Авторы.Код

GROUP BY [Автор книги].[Код книги]) AS Авторы ON Книга.Код = Авторы.[Код книги]

INNER JOIN

Стеллажи ON Книга.[Код стеллажа] = Стеллажи.Код

INNER JOIN

Помещения ON Стеллажи.[Код зала] = Помещения.Код

WHERE

Книга.Название LIKE '%' + @sterm + '%' OR

Авторы.Автор LIKE '%' + @sterm + '%' OR

EXISTS (

SELECT 1

FROM

Жанр

WHERE

Жанр.[Название жанра] LIKE '%' + @sterm + '%'

)

GROUP BY Книга.Название, Авторы.Автор, Стеллажи.[Номер стеллажа], Помещения.Название, Помещения.Название, Помещения.[Адрес помещения]

END;

GO

--КОНЕЦ ОТЧЕТОВ

GO

-- ОПЕРАЦИИ

CREATE PROCEDURE RemoveOutdatedReaders AS

BEGIN

INSERT INTO АрхивЧитателей ([Фамилия], [Имя], [Отчество], [Дата рождения],

[Контактный номер], [Адрес проживания], [Данные паспорта],

[Номер читательского билета], [Название группы], [Факультет],

[Задолженности], [Имя для входа])

SELECT Читатель.[Фамилия], Читатель.[Имя], Читатель.[Отчество], Читатель.[Дата рождения],

Читатель.[Контактный номер], Читатель.[Адрес проживания], Читатель.[Данные паспорта],

Читатель.[Номер читательского билета], Группа.[Название], Факультет.[Название],

(SELECT STRING\_AGG(CONCAT(Книга.[Название], ' (', [Выдача и возврат].[Инвентарный номер], ')', ' - ',

CONVERT(VARCHAR(10), [Дата выдачи], 104)), ', ')

FROM [Выдача и возврат]

INNER JOIN Книга ON [Выдача и возврат].[Код книги] = Книга.[Код]

WHERE [Выдача и возврат].[Код читателя] = Читатель.[Код] AND [Дата возврата] IS NULL) AS [Задолженности],

Читатель.[Имя для входа]

FROM Читатель

INNER JOIN Группа ON Читатель.[Код группы] = Группа.Код

INNER JOIN Факультет ON Группа.[Код факультета] = Факультет.Код

WHERE YEAR(GETDATE()) > Группа.[Год окончания];

DELETE FROM [Выдача и возврат] where [Код читателя] IN(SELECT [Код] FROM АрхивЧитателей)

DELETE FROM Читатель WHERE [Код] IN (SELECT [Код] FROM АрхивЧитателей);

END;

GO

--КОНЕЦ ОПЕРАЦИИ

GO

# 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 LibA

{

public partial class UserPanel : Form

{

private AuthForm authForm;

public UserPanel()

{

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

this.Refresh();

this.AutoSize = false;

InitializeComponent();

Screen screen = Screen.FromControl(this);

this.Location = new Point(screen.WorkingArea.Right - this.Width, screen.WorkingArea.Bottom - this.Height);

ConnectionManager.Instance.Disconnection += (sender, e) =>

{

администрированиеToolStripMenuItem.Visible = false;

};

}

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 зарегистрироватьсяToolStripMenuItem\_Click(object sender, EventArgs e)

{

authForm = new AuthForm(this);

authForm.RegAuthSuccess += ChangeStatus;

authForm.WhichWindow(WindowType.REGISTER);

authForm.ShowDialog();

}

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

{

authForm = new AuthForm(this);

authForm.RegAuthSuccess += ChangeStatus;

authForm.HasRights += (sender, e) =>

{

администрированиеToolStripMenuItem.Visible = true;

};

authForm.WhichWindow(WindowType.AUTHORIZE);

authForm.ShowDialog();

}

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

{

Environment.Exit(0);

}

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

{

ConnectionManager.Instance.Disconnect();

this.statText.Text = "";

ConnectionManager.Instance.SetupConnectionString("Guest1", "1");

}

public void newTSSLabel(string ctext)

{

this.statText.Text = ctext;

}

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

{

Program.MakeFocus(SettingsPane.Instance);

}

private void выходToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Close();

Dispose();

}

private void администрированиеToolStripMenuItem\_Click(object sender = null, EventArgs e = null)

{

Program.MakeFocus(AdminPanel.Instance);

}

private async void IsDBAliveTimer\_Tick(object sender, EventArgs e)

{

try

{

bool isDBAlive = await ConnectionManager.CheckDBConnectionAsync();

if (isDBAlive)

{

this.DBStat.Image = Properties.Resources.ok;

this.DBStat.Text = "База данных: ОК";

IsDBAliveTimer.Interval = 60 \* 1000;

}

else

{

this.DBStat.Image = Properties.Resources.error;

this.DBStat.Text = "База данных: ОШИБКА";

IsDBAliveTimer.Interval = 10 \* 1000;

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при проверке базы данных: {ex.Message}");

}

}

private async void ChangeStatus(object sender, EventArgs e)

{

using (SqlConnection c = await ConnectionManager.Instance.OpenConnection())

{

var b = new SqlConnectionStringBuilder(c?.ConnectionString);

statText.Text = $"Добро пожаловать, {b?.UserID}";

}

}

private void UserPanel\_SizeChanged(object sender = null, EventArgs e = null)

{

int leftMargin = (int)(Width \* 0.1);

int topMargin = (int)(Height \* 0.05);

int rightMargin = (int)(Width \* 0.1);

int topMarginWithMenuStrip = topMargin + menuStrip1.Height;

searchPanel.Location = new Point(leftMargin, topMarginWithMenuStrip);

searchPanel.Width = Width - leftMargin - rightMargin;

UpdateSearchPanelLayout();

}

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

{

if (searchInput.Text == "Начните вводить что-нибудь...")

{

searchInput.Text = "";

searchInput.ForeColor = SystemColors.WindowText;

UpdateSearchPanelLayout();

}

}

private void searchInput\_Leave(object sender, EventArgs e)

{

if (string.IsNullOrWhiteSpace(searchInput.Text))

{

searchInput.Text = "Начните вводить что-нибудь...";

searchInput.ForeColor = SystemColors.GrayText;

UpdateSearchPanelLayout();

}

}

private void UpdateSearchPanelLayout()

{

int panelWidth = searchPanel.Width;

int panelHeight = searchPanel.Height;

int doSearchSize = panelHeight;

doSearch.Size = new Size(doSearchSize, doSearchSize);

doSearch.Location = new Point(panelWidth - doSearchSize, 0);

int searchInputWidth = panelWidth - doSearchSize;

int searchInputHeight = panelHeight;

searchInput.Size = new Size(searchInputWidth, searchInputHeight);

searchInput.Location = new Point(0, 0);

float fontSize = searchInputHeight \* 0.7f;

if (fontSize > searchInputHeight)

fontSize = searchInputHeight;

Font font = new Font("Arial", fontSize);

searchInput.Font = font;

searchInput.TextAlign = HorizontalAlignment.Left;

searchInput.Select(0, 0);

int dataGridViewMargin = 10;

int dataGridViewTopMargin = searchPanel.Bottom + dataGridViewMargin;

dataGridViewMain.Location = new Point(searchPanel.Left, searchPanel.Bottom + 10);

dataGridViewMain.Width = searchPanel.Width;

dataGridViewMain.Height = this.ClientSize.Height - dataGridViewMain.Top - statusStrip1.Height;

}

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

{

using (SqlCommand command = new SqlCommand("SELECT [Название],[Автор],[Жанр],[Издательство],[Год выпуска],[Число страниц],[Язык книги],[Доступность],[Как часто брали],[Краткое описание] FROM BooksPublicInfo(@SearchTerm)", await ConnectionManager.Instance.OpenConnection()))

{

try

{

if (searchInput.Text == string.Empty || searchInput.Text.Equals("Начните вводить что-нибудь..."))

throw new Exception();

command.Parameters.AddWithValue("@SearchTerm", searchInput.Text);

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

//dataTable.Columns.Remove("Обложка");

string[] availability = new string[dataTable.Rows.Count];

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

{

availability[i] = ((int)dataTable.Rows[i]["Доступность"] > 0) ? "В наличии" : "Нет в наличии";

}

int columnIndex = dataTable.Columns["Доступность"].Ordinal;

dataTable.Columns.Remove("Доступность");

dataTable.Columns.Add("Доступность", typeof(string));

dataTable.Columns["Доступность"].SetOrdinal(columnIndex);

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

{

dataTable.Rows[i]["Доступность"] = availability[i];

}

dataGridViewMain.DataSource = dataTable;

dataGridViewMain.Visible = true;

dataGridViewMain.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

foreach (DataGridViewRow row in dataGridViewMain.Rows)

{

foreach (DataGridViewCell cell in row.Cells)

{

if (cell.Value != null)

{

cell.Style.WrapMode = DataGridViewTriState.True;

}

}

}

}

catch

{

dataGridViewMain.Visible = false;

}

}

}

private void dataGridViewMain\_DataError(object sender, DataGridViewDataErrorEventArgs e)

{

}

private void dataGridViewMain\_CellFormatting(object sender, DataGridViewCellFormattingEventArgs e)

{

if (dataGridViewMain.Columns[e.ColumnIndex].Name == "Доступность")

{

if (e.Value != null && e.Value == "В наличии")

{

e.CellStyle.BackColor = Color.LightGreen;

}

else

e.CellStyle.BackColor = Color.FromArgb(255, 255, 192, 192);

}

}

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace LibA

{

public enum ReportType{

With2Calendars,

TextInput,

None

}

public partial class Reports : Form

{

string sqlCommand;

public Reports()

{

InitializeComponent();

}

private void Reports\_SizeChanged(object sender, EventArgs e)

{

if (dateTimePicker2.Right > b\_with2Calendars.Left)

{

this.Width = dateTimePicker2.Right + b\_with2Calendars.Width + 20;

}

}

public async void MakeReport(string sqlcommand, ReportType rtype)

{

this.sqlCommand = sqlcommand;

if (rtype is ReportType.With2Calendars)

{

gbox\_with2Calendars.Visible = true;

gbox\_with2Calendars.Dock = DockStyle.Top;

}

else if (rtype is ReportType.TextInput) {

gbox\_WithTextInput.Visible = true;

gbox\_WithTextInput.Dock = DockStyle.Top;

i\_sterm.Width = b\_sterm.Left;

}

else if (rtype is ReportType.None)

{

DataTable dt = new();

using (SqlCommand comm = new SqlCommand(this.sqlCommand, await ConnectionManager.Instance.OpenConnection()))

{

dt = DBWorker.GetDataTable(comm);

dataGridViewMain.DataSource = dt;

}

}

dataGridViewMain.Visible = true;

dataGridViewMain.Dock = DockStyle.Fill;

}

private async void b\_with2Calendars\_Click(object sender, System.EventArgs e)

{

DateTime date1 = dateTimePicker1.Value;

DateTime date2 = dateTimePicker2.Value;

if (date1 <= date2)

{

string startDate = dateTimePicker1.Value.ToString("yyyy-MM-dd");

string endDate = dateTimePicker2.Value.ToString("yyyy-MM-dd");

DataTable dt = new();

dt = await DBWorker.GetDataTable(this.sqlCommand,startDate,endDate);

dataGridViewMain.DataSource = dt;

}

}

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

{

DataTable dt = new();

dt = await DBWorker.GetDataTable(this.sqlCommand, i\_sterm.Text);

dataGridViewMain.DataSource = dt;

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Windows.Forms;

namespace LibA

{

public partial class GiveBook : Form

{

public GiveBook()

{

InitializeComponent();

// Инициализация формы

InitializeForm();

}

private void InitializeForm()

{

// Загрузка данных о факультетах

LoadFaculties();

// Установка связей между элементами управления

listBoxFaculties.SelectedIndexChanged += ListBoxFaculties\_SelectedIndexChanged;

listBoxGroups.SelectedIndexChanged += ListBoxGroups\_SelectedIndexChanged;

}

private async void LoadFaculties()

{

DataTable facultyTable = await DBWorker.GetDataTable("Факультет");

if (facultyTable != null)

{

listBoxFaculties.DataSource = facultyTable;

listBoxFaculties.DisplayMember = "Название";

}

}

private void ListBoxFaculties\_SelectedIndexChanged(object sender, EventArgs e)

{

if (listBoxFaculties.SelectedIndex != -1)

{

DataRowView selectedFaculty = (DataRowView)listBoxFaculties.SelectedItem;

LoadGroups(selectedFaculty["Название"].ToString());

}

}

private async void LoadGroups(string v)

{

string sqlCommand = $"SELECT \* FROM Группа WHERE [Код факультета] = (select Код from Факультет where [Название] = '{v}')";

DataTable groupTable = await DBWorker.GetDataTable(sqlCommand);

if (groupTable != null)

{

listBoxGroups.DataSource = groupTable;

listBoxGroups.DisplayMember = "Название";

}

}

private void ListBoxGroups\_SelectedIndexChanged(object sender, EventArgs e)

{

if (listBoxGroups.SelectedIndex != -1)

{

DataRowView selectedGroup = (DataRowView)listBoxGroups.SelectedItem;

LoadReaders(selectedGroup["Название"].ToString());

}

}

private async void LoadReaders(string v)

{

DataTable readerTable = await DBWorker.GetDataTable($"select \* from Читатель where [Код группы] in (select Код from Группа where [Название] = '{v}')");

if (readerTable != null)

{

dataGridView1.DataSource = readerTable;

}

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

if (column.Name.ToLower().StartsWith("код") || column.Name.ToLower().StartsWith("id"))

{

column.Visible = false;

}

}

}

private void dataGridView1\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int readerCode = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["Код"].Value);

LoadIssueReturn(readerCode);

}

}

private async void LoadIssueReturn(int readerCode)

{

string sqlCommand = "SELECT TOP 0 \* FROM [Выдача и возврат]";

DataTable issueReturnTable = await DBWorker.GetDataTable(sqlCommand);

if (issueReturnTable != null)

{

issueReturnTable.Columns.Remove("Код");

issueReturnTable.Columns["Код читателя"].DefaultValue = readerCode;

issueReturnTable.TableName = "Выдача и возврат";

button1.Visible = true;

button2.Visible = true;

dataGridView2.Visible = true;

dataGridView2.Dock = DockStyle.Fill;

dataGridView2.DataSource = issueReturnTable;

DataRow newRow = issueReturnTable.NewRow();

//newRow["Код читателя"] = readerCode;

issueReturnTable.Rows.Add(newRow);

}

}

private async void dataGridView2\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.RowIndex >= 0 && e.ColumnIndex >= 0)

{

string headerText = dataGridView2.Columns[e.ColumnIndex].HeaderText.ToLower();

if (headerText.StartsWith("код") || headerText.StartsWith("id"))

{

string columnName = dataGridView2.Columns[e.ColumnIndex].Name;

string ctableName = "Выдача и возврат";

Console.WriteLine($"{columnName}, {ctableName}");

List<string> tableNames = await DBWorker.GetLinkedTableNames(ctableName, columnName);

if (tableNames.Count > 0)

{

foreach (string tableName in tableNames)

{

DataTable dependentTable = await DBWorker.GetDataTable(tableName);

if (dependentTable != null)

{

string cellValueString = dataGridView2.Rows[e.RowIndex].Cells[e.ColumnIndex].Value?.ToString();

int val = string.IsNullOrEmpty(cellValueString) ? 0 : int.Parse(cellValueString);

DLinker dLinkerForm = new DLinker(dependentTable, val);

dLinkerForm.SelectionMade += (selectedValue) =>

{

dataGridView2.EndEdit();

dataGridView2.Rows[e.RowIndex].Cells[e.ColumnIndex].Value = selectedValue;

};

dLinkerForm.ShowDialog();

}

else

{

MessageBox.Show($"Связанная таблица '{tableName}' не найдена.");

}

}

}

else

{

MessageBox.Show("Связанная таблица не найдена.");

}

}

}

}

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

{

dataGridView2.DataSource = null;

dataGridView2.Visible = false;

button1.Visible = false;

button2.Visible = false;

}

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

{

try

{

DataTable changes = ((DataTable)dataGridView2.DataSource).GetChanges();

if (changes != null)

{

await DBWorker.BeginTransaction(changes);

((DataTable)dataGridView2.DataSource).AcceptChanges();

}

}

catch (Exception ex) {

MessageBox.Show("Проверьте правильность заполнения и попробуйте ещё раз", "Ошибка при заполнении данных");

Console.WriteLine(ex);

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Controls;

using System.Windows.Forms;

namespace LibA

{

public partial class DLinker : Form

{

public delegate void SelectionDelegate(string selectedValue);

public event SelectionDelegate SelectionMade;

public DLinker(DataTable dt, int code = 0)

{

InitializeComponent();

dataGridViewMain.DataSource = dt;

dataGridViewMain.Columns[0].Visible = false;

this.Text = $"Режим выбора: {dt.TableName}";

this.Load += (sender, e) =>

{

DataGridViewRow row = dataGridViewMain.Rows

.Cast<DataGridViewRow>()

.FirstOrDefault(r => Convert.ToInt32(r.Cells["Код"].Value) == code);

if (row != null)

{

int rowIndex = row.Index;

dataGridViewMain.FirstDisplayedScrollingRowIndex = rowIndex;

row.Selected = true;

}

};

}

private async void dataGridViewMain\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex >= 0 && e.ColumnIndex >= 0)

{

string headerText = dataGridViewMain.Columns[e.ColumnIndex].HeaderText.ToLower();

if ((headerText.StartsWith("код") || headerText.StartsWith("id")) && !(headerText.Equals("код", StringComparison.OrdinalIgnoreCase) || headerText.Equals("id", StringComparison.OrdinalIgnoreCase)))

{

string columnName = dataGridViewMain.Columns[e.ColumnIndex].Name;

string ctableName = ((DataTable)dataGridViewMain.DataSource)?.TableName;

List<string> tableNames = await DBWorker.GetLinkedTableNames(ctableName, columnName);

if (tableNames.Count > 0)

{

foreach (string tableName in tableNames)

{

DataTable dependentTable = await DBWorker.GetDataTable(tableName);

if (dependentTable != null)

{

DLinker dLinkerForm = new DLinker(dependentTable, (int)dataGridViewMain.Rows[e.RowIndex].Cells[e.ColumnIndex].Value);

dLinkerForm.SelectionMade += (selectedValue) =>

{

dataGridViewMain.EndEdit();

dataGridViewMain.Rows[e.RowIndex].Cells[e.ColumnIndex].Value = selectedValue;

};

dLinkerForm.ShowDialog();

}

else

{

MessageBox.Show($"Связанная таблица '{tableName}' не найдена.");

}

}

}

else

{

MessageBox.Show("Связанная таблица не найдена.");

}

}

}

else

{

string selectedValue = dataGridViewMain.Rows[e.RowIndex].Cells[0].Value.ToString();

SelectionMade?.Invoke(selectedValue);

this.Close();

}

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.IO.Packaging;

using System.Linq;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace LibA

{

internal class DBWorker

{

private static DataTable oldTable;

public static DataTable OldTable

{

set { oldTable = value; }

get { return oldTable!; }

}

public static async Task<string[]> BdGetDataMSSQL(string commandText)

{

List<string> result = new List<string>();

try

{

using (SqlConnection connection = await ConnectionManager.Instance.OpenConnection())

{

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

{

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

{

while (await reader.ReadAsync())

{

result.Add(reader.GetString(0));

}

}

}

connection.Close();

}

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

throw;

}

return result.ToArray();

}

public static DataTable GetDataTable(SqlCommand command) {

using (SqlDataAdapter adapter = new SqlDataAdapter(command))

{

DataTable dataTable = new();

adapter.Fill(dataTable);

return dataTable;

}

}

public static async Task<DataTable> GetDataTable(string tableName)

{

string commandText = null;

if (tableName.ToLower().StartsWith("select"))

commandText = tableName;

else

commandText = $"SELECT \* FROM [{tableName}]";

try

{

using (SqlCommand command = new SqlCommand(commandText, await ConnectionManager.Instance.OpenConnection()))

{

using (SqlDataAdapter adapter = new SqlDataAdapter(command))

{

DataTable dataTable = new(tableName);

adapter.Fill(dataTable);

return dataTable;

}

}

}

catch (Exception e)

{

MessageBox.Show(e.ToString());

}

return null;

}

public static async Task ExecProcedure(string procname, params object[] parameters) {

try

{

if (procname.ToLower().StartsWith("exec"))

procname = "Execute " + procname;

using (SqlCommand command = new SqlCommand(procname, await ConnectionManager.Instance.OpenConnection()))

{

Regex regex = new Regex(@"@\w+");

MatchCollection matches = regex.Matches(procname);

if (matches.Count != parameters.Length)

{

throw new ArgumentException("Количество параметров не соответствует количеству имен параметров в запросе.");

}

for (int i = 0; i < parameters.Length; i++)

{

string paramName = matches[i].Value;

command.Parameters.AddWithValue(paramName, parameters[i]);

}

await command.ExecuteNonQueryAsync();

}

}

catch (Exception e)

{

MessageBox.Show($"Ошибка выполнения процедуры {procname}: {e.Message}");

}

}

public static async Task<DataTable> GetDataTable(string procname, params object[] parameters)

{

try

{

if (procname.ToLower().StartsWith("exec"))

procname = "Execute " + procname;

using (SqlCommand command = new SqlCommand(procname, await ConnectionManager.Instance.OpenConnection()))

{

Regex regex = new Regex(@"@\w+");

MatchCollection matches = regex.Matches(procname);

if (matches.Count != parameters.Length)

{

throw new ArgumentException("Количество параметров не соответствует количеству имен параметров в запросе.");

}

for (int i = 0; i < parameters.Length; i++)

{

string paramName = matches[i].Value;

command.Parameters.AddWithValue(paramName, parameters[i]);

}

using (SqlDataAdapter adapter = new SqlDataAdapter(command))

{

DataTable dt = new DataTable();

adapter.Fill(dt);

return dt;

}

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка в {procname}: {ex.Message}");

return null;

}

}

public static async Task BeginTransaction(DataGridView dgv)

{

DataTable dataTable = dgv.DataSource as DataTable;

oldTable = dataTable;

await BeginTransaction(dataTable);

}

public static async Task BeginTransaction(DataTable dt)

{

using (SqlConnection connection = await ConnectionManager.Instance.OpenConnection())

{

IEnumerable<string> userPermissions = await CheckTablePermissions(dt.TableName, connection);

if (!userPermissions.Except(new[] { "SELECT" }, StringComparer.OrdinalIgnoreCase).Any())

{

MessageBox.Show("У вас нет прав на выполнение операций в данной таблице.");

throw new Exception();

}

using (SqlTransaction transaction = connection.BeginTransaction())

{

if (dt != null)

{

try

{

using (SqlDataAdapter adapter = new SqlDataAdapter($"SELECT \* FROM [{dt.TableName}]", connection))

{

adapter.SelectCommand.Transaction = transaction;

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);

if (userPermissions.Contains("INSERT", StringComparer.OrdinalIgnoreCase))

adapter.InsertCommand = commandBuilder.GetInsertCommand();

if (userPermissions.Contains("UPDATE", StringComparer.OrdinalIgnoreCase))

adapter.UpdateCommand = commandBuilder.GetUpdateCommand();

if (userPermissions.Contains("DELETE", StringComparer.OrdinalIgnoreCase))

adapter.DeleteCommand = commandBuilder.GetDeleteCommand();

adapter.Update(dt);

transaction.Commit();

}

}

catch

{

transaction.Rollback();

//MessageBox.Show($"Ошибка при обновлении базы данных");

throw;

}

}

}

}

}

public static async Task RollbackTransaction()

{

try

{

await BeginTransaction(oldTable);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при транзакции: {ex.Message}");

}

}

public static async Task<List<string>> GetLinkedTableNames(string parentTableName, string columnName)

{

List<string> linkedTables = new List<string>();

try

{

using (SqlConnection connection = await ConnectionManager.Instance.OpenConnection())

{

SqlCommand command = new SqlCommand("GetDependentTableName", connection);

command.CommandType = CommandType.StoredProcedure;

command.Parameters.AddWithValue("@ParentTableName", parentTableName);

command.Parameters.AddWithValue("@ColumnName", columnName);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

string referencedTable = reader[0].ToString();

linkedTables.Add(referencedTable);

}

reader.Close();

}

}

catch

{

}

return linkedTables;

}

private static async Task<IEnumerable<string>> CheckTablePermissions(string tableName, SqlConnection connection)

{

//TODO

var builder = new SqlConnectionStringBuilder(connection.ConnectionString);

string currentUsername = builder.UserID;

if (string.IsNullOrEmpty(currentUsername))

{

MessageBox.Show("Не удалось определить текущего пользователя.");

return Enumerable.Empty<string>();

}

List<string> userPermissions = new List<string>();

try

{

if (await CheckUserRights(connection))

return new List<string> { "SELECT", "INSERT", "UPDATE", "DELETE", "ALTER", "CREATE", "DROP", "EXECUTE", "GRANT", "REFERENCES", "VIEW DEFINITION" };

string getPermissionsQuery = "EXEC GetUserTablePermissions @DatabaseUserName, @TableName";

using (SqlCommand permissionsCommand = new SqlCommand(getPermissionsQuery, connection))

{

permissionsCommand.Parameters.AddWithValue("@DatabaseUserName", currentUsername);

permissionsCommand.Parameters.AddWithValue("@TableName", tableName);

using (SqlDataReader permissionsReader = await permissionsCommand.ExecuteReaderAsync())

{

while (permissionsReader.Read())

{

string permission = permissionsReader.GetString(0);

userPermissions.Add(permission);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при проверке прав пользователя: {ex.Message}");

return Enumerable.Empty<string>();

}

return userPermissions;

}

public static async Task<bool> CheckUserRights(SqlConnection connection)

{

//TODO

var builder = new SqlConnectionStringBuilder(connection.ConnectionString);

string userID = builder.UserID;

string getRolesQuery = "EXEC GetUserRoles @LoginName";

using (SqlCommand rolesCommand = new SqlCommand(getRolesQuery, connection))

{

rolesCommand.Parameters.AddWithValue("@LoginName", userID);

using (SqlDataReader rolesReader = await rolesCommand.ExecuteReaderAsync())

{

while (rolesReader.Read())

{

string roleName = rolesReader.GetString(0);

if (Enum.TryParse<DatabaseRoles>(roleName, out DatabaseRoles role))

{

if (role != DatabaseRoles.None)

{

return true;

}

}

}

return false;

}

}

}

public enum DatabaseRoles

{

None,

db\_securityadmin,

db\_owner,

db\_denydatawriter,

db\_denydatareader,

db\_ddladmin,

db\_datawriter,

db\_datareader,

db\_backupoperator,

db\_accessadmin,

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

}

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.IO;

using System.Net.Sockets;

using System.Security.Cryptography;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace LibA {

public interface IConnectionManager

{

void Disconnect();

bool SetupConnectionString(string login = null, string password = null);

Task<SqlConnection> OpenConnection();

Task SendRegData(string name, string login, string password);

string ReceiveResponce(TcpClient tcpClient);

}

public class ConnectionManager

{

private static ConnectionManager \_instance;

private String connectionString;

private const String CRYPTOKEY = "ThisIsASecretKey1234567890123456";

public event EventHandler Disconnection;

private ConnectionManager() { }

public static ConnectionManager Instance

{

get

{

if (\_instance is null)

\_instance = new ConnectionManager();

return \_instance;

}

}

public void Disconnect()

{

if (\_instance != null)

{

\_instance.connectionString = null;

SetupConnectionString("Guest1", "1");

DBWorker.OldTable = null;

Disconnection?.Invoke(this, EventArgs.Empty);

}

}

public Boolean SetupConnectionString(string login = null, string password = null)

{

string backup = connectionString;

connectionString = $"Data Source={Properties.Settings.Default.dbConnSourceAddr}{(Properties.Settings.Default.DBPort == "" ? "" : $",{Properties.Settings.Default.DBPort}")};Database={Properties.Settings.Default.ICatalog};";

if (login is null || password is null)

{

MessageBox.Show("Поля не могут быть пусты");

return false;

}

else

connectionString += $"User Id = {login};Password='{password}'";

try

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

if (conn.State is not ConnectionState.Open)

{

throw new Exception();

}

}

}

catch (Exception e)

{

connectionString = backup;

MessageBox.Show("Ошибка создания соединения. Проверьте данные и попробуйте ещё раз");

Console.WriteLine(e.Message);

return false; //not valid

}

return true; //valid

}

public async Task<SqlConnection> OpenConnection()

{

SqlConnection connection = null;

try

{

connection = new SqlConnection(connectionString);

await Task.Run(() => connection.OpenAsync());

return connection;

}

catch (SqlException ex)

{

MessageBox.Show("Ошибка установки пользователя: " + ex.Message);

connection?.Dispose();

}

catch

{

MessageBox.Show("Ошибка открытия соединения\n \* Проверьте, включен ли сервер\n \* Пользователь авторизован? ");

connection?.Dispose();

}

return null;

}

public static async Task<bool> CheckDBConnectionAsync()

{

try

{

using (TcpClient tcpClient = new TcpClient())

{

await tcpClient.ConnectAsync(Properties.Settings.Default.dbConnSourceAddr, short.Parse(Properties.Settings.Default.DBPort));

return true;

}

}

catch

{

return false;

}

}

public async Task<string> SendRegData(string name, string login, string password)

{

string response = null;

TcpClient tcpClient = new TcpClient();

try

{

await tcpClient.ConnectAsync(Properties.Settings.Default.dbConnSourceAddr, 8888);

string credentials = $"{name},{EncryptString(login)},{EncryptString(password)}";

NetworkStream networkStream = tcpClient.GetStream();

byte[] bytes = Encoding.UTF8.GetBytes(credentials);

await networkStream.WriteAsync(bytes, 0, bytes.Length);

response = await ReceiveResponceAsync(tcpClient);

}

catch (Exception)

{

MessageBox.Show("Ошибка регистрации");

}

finally

{

tcpClient.Close();

}

return response;

}

public async Task<string> ReceiveResponceAsync(TcpClient tcpClient)

{

NetworkStream networkStream = tcpClient.GetStream();

byte[] buffer = new byte[1024];

StringBuilder responseBuilder = new StringBuilder();

try

{

int bytesRead;

do

{

bytesRead = await networkStream.ReadAsync(buffer, 0, buffer.Length);

if (bytesRead > 0)

{

responseBuilder.Append(Encoding.UTF8.GetString(buffer, 0, bytesRead));

}

} while (bytesRead > 0);

}

catch (Exception)

{

// Обработка исключений

}

return responseBuilder.ToString();

}

private string EncryptString(string plainText)

{

using (Aes aesAlg = Aes.Create())

{

aesAlg.Key = Encoding.UTF8.GetBytes(CRYPTOKEY);

aesAlg.IV = new byte[aesAlg.BlockSize / 8];

ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);

using (MemoryStream msEncrypt = new MemoryStream())

{

using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))

using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))

{

swEncrypt.Write(plainText);

}

return Convert.ToBase64String(msEncrypt.ToArray());

}

}

}

/\*private string DecryptString(string cipherText)

{

using (Aes aesAlg = Aes.Create())

{

aesAlg.Key = Encoding.UTF8.GetBytes(CRYPTOKEY);

aesAlg.IV = new byte[aesAlg.BlockSize / 8];

ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);

using (MemoryStream msDecrypt = new MemoryStream(Convert.FromBase64String(cipherText)))

{

using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))

using (StreamReader srDecrypt = new StreamReader(csDecrypt))

{

return srDecrypt.ReadToEnd();

}

}

}

}

\*/

}

}

using OfficeOpenXml;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Globalization;

using System.IO;

using System.Linq;

using System.Runtime.Remoting;

using System.Threading.Tasks;

using System.Windows.Controls;

using System.Windows.Controls.Primitives;

using System.Windows.Forms;

namespace LibA

{

public partial class AdminPanel : Form

{

private static AdminPanel instance;

private static bool isClosed = true;

public static AdminPanel Instance

{

get

{

if (instance == null || isClosed)

{

instance = new AdminPanel();

instance.FormClosed += (sender, e) =>

{

isClosed = true;

};

isClosed = false;

}

return instance;

}

}

private AdminPanel()

{

InitializeComponent();

this.Shown += LoadTablesAsync;

}

private async void LoadTablesAsync(object sender, EventArgs e)

{

string[] tables = null;

try

{

tables = await DBWorker.BdGetDataMSSQL("SELECT table\_name FROM information\_schema.tables WHERE table\_type = 'BASE TABLE' AND table\_name != 'sysdiagrams' order by table\_name ASC");

}

catch

{

this.Close();

return;

}

if (tables != null)

{

Tables.Items.AddRange(tables);

}

Tables.Height = Tables.ItemHeight \* (Tables.Items.Count + 1);

buttonTransact.Location = new Point(

Tables.Left,

Tables.Bottom

);

buttonRollback.Location = new Point(

buttonTransact.Left,

buttonTransact.Bottom

);

this.Height = dataGridViewMain.Bottom;

this.Width = dataGridViewMain.Right + (this.Width - this.ClientSize.Width);

}

private async void Tables\_SelectedIndexChanged(object sender, EventArgs e)

{

if (DBWorker.OldTable != null && !AreTablesEqual(dataGridViewMain.DataSource as DataTable, DBWorker.OldTable))

{

DialogResult dgv = MessageBox.Show("У вас есть несохранённые изменения. Вы хотите сохранить их?", "Предупреждение", MessageBoxButtons.YesNo, MessageBoxIcon.Hand);

if (dgv == DialogResult.Yes)

{

buttonTransact\_Click(sender, e);

return;

}

}

string selectedTable = Tables.Items[Tables.SelectedIndex > 0 ? Tables.SelectedIndex : 0].ToString();

DataTable datatable = await DBWorker.GetDataTable(selectedTable);

if (datatable != null)

{

buttonRollback.Enabled = false;

buttonRollback.BackColor = Color.Gray;

DBWorker.OldTable = datatable.Copy();

dataGridViewMain.DataSource = datatable;

dataGridViewMain.Enabled = true;

dataGridViewMain.Visible = true;

dataGridViewMain.Columns[0].Visible = false;

dataGridViewMain.AutoGenerateColumns = true;

}

}

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

{

this.Close();

this.Dispose();

}

private bool AreTablesEqual(DataTable table1, DataTable table2)

{

try

{

if (table1 == null || table2 == null)

{

return table1 == table2;

}

if (table1.Rows.Count != table2.Rows.Count || table1.Columns.Count != table2.Columns.Count)

{

return false;

}

for (int i = 0; i < table1.Rows.Count; i++)

{

for (int j = 0; j < table1.Columns.Count; j++)

{

if (!object.Equals(table1.Rows[i][j], table2.Rows[i][j]))

{

return false;

}

}

}

}

catch

{

return false;

}

return true;

}

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

{

try

{

await DBWorker.BeginTransaction(dataGridViewMain);

buttonRollback.Enabled = true;

buttonRollback.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(255)))), ((int)(((byte)(128)))), ((int)(((byte)(128)))));

MessageBox.Show("Изменения сохранены в базе данных.");

}

catch (Exception er)

{

MessageBox.Show("Непредвиденная ошибка при обновлении данных:\n" + er.Message);

}

}

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

{

try

{

dataGridViewMain.DataSource = DBWorker.OldTable;

await DBWorker.BeginTransaction(DBWorker.OldTable);

buttonRollback.Enabled = false;

buttonRollback.BackColor = Color.Gray;

MessageBox.Show("Откат выполнен успешно");

}

catch

{

MessageBox.Show("Ошибка отката");

}

}

private void dataGridViewMain\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

try

{

if (dataGridViewMain.Columns[e.ColumnIndex] is DataGridViewTextBoxColumn column &&

column.DefaultCellStyle.Format == "dd.MM.yyyy")

{

if (DateTime.TryParseExact(

dataGridViewMain.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString(),

"dd.MM.yyyy",

CultureInfo.InvariantCulture,

DateTimeStyles.None,

out DateTime date))

{

dataGridViewMain.Rows[e.RowIndex].Cells[e.ColumnIndex].Value = date.ToString("dd.MM.yyyy");

}

else

{

MessageBox.Show("Неверный формат даты!");

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при редактировании: {ex.Message}");

}

}

private async void lPane\_Panel2\_Click(object sender, EventArgs e)

{

int targetWidth;

if (lPane.Panel1Collapsed)

{

targetWidth = 200;

lPane.Panel1Collapsed = !lPane.Panel1Collapsed;

await AnimateSplitter(targetWidth);

}

else

{

targetWidth = 40;

await AnimateSplitter(targetWidth);

lPane.Panel1Collapsed = !lPane.Panel1Collapsed;

}

lPane.Panel2.BackgroundImage.RotateFlip(RotateFlipType.RotateNoneFlipX);

lPane.Refresh();

}

private async Task AnimateSplitter(int targetWidth)

{

const int AnimationDuration = 100;

const int AnimationSteps = 10;

int currentWidth = lPane.Width;

int step = (targetWidth - currentWidth) / AnimationSteps;

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

{

currentWidth += step;

lPane.Width = currentWidth;

await Task.Delay(AnimationDuration / AnimationSteps);

}

lPane.Width = targetWidth;

dataGridViewMain.Dock = DockStyle.Fill;

}

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

{

Program.MakeFocus(SettingsPane.Instance);

}

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

{

ConnectionManager.Instance.Disconnect();

this.Close();

}

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

{

this.Close();

}

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

{

Reports dln = new Reports();

dln.MakeReport("GetPercentageUsersByGroupAndFaculty @startdate, @enddate", ReportType.With2Calendars);

ToolStripMenuItem menuItem = sender as ToolStripMenuItem;

dln.Text = menuItem.Text;

dln.ShowDialog();

}

private void debtorsCommonBetween2DatesToolStripMenuItem\_Click(object sender, EventArgs e)

{

Reports dln = new Reports();

dln.MakeReport("GetDebtorsReport @startdate, @enddate", ReportType.With2Calendars);

ToolStripMenuItem menuItem = sender as ToolStripMenuItem;

dln.Text = menuItem.Text;

dln.ShowDialog();

}

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

{

using (Reports dln = new()) {

dln.MakeReport("GetReaderInfoExact", ReportType.None);

ToolStripMenuItem menuItem = sender as ToolStripMenuItem;

dln.Text = menuItem.Text;

dln.ShowDialog();

}

}

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

{

using (Reports dln = new())

{

dln.MakeReport("FindBookLocation @sterm", ReportType.TextInput);

ToolStripMenuItem menuItem = sender as ToolStripMenuItem;

dln.Text = menuItem.Text;

dln.ShowDialog();

}

}

//OPERATIONS

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

{

await DBWorker.ExecProcedure("RemoveOutdatedReaders");

MessageBox.Show("Выполнение процедуры успешно");

}

private async void dataGridViewMain\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.RowIndex >= 0 && e.ColumnIndex >= 0)

{

string headerText = dataGridViewMain.Columns[e.ColumnIndex].HeaderText.ToLower();

if (headerText.StartsWith("код") || headerText.StartsWith("id"))

{

string columnName = dataGridViewMain.Columns[e.ColumnIndex].Name;

string ctableName = ((DataTable)dataGridViewMain.DataSource)?.TableName;

List<string> tableNames = await DBWorker.GetLinkedTableNames(ctableName, columnName);

if (tableNames.Count > 0)

{

foreach (string tableName in tableNames)

{

DataTable dependentTable = await DBWorker.GetDataTable(tableName);

if (dependentTable != null)

{

string cellValueString = dataGridViewMain.Rows[e.RowIndex].Cells[e.ColumnIndex].Value?.ToString();

int val = string.IsNullOrEmpty(cellValueString) ? 0 : int.Parse(cellValueString);

DLinker dLinkerForm = new DLinker(dependentTable, val);

dLinkerForm.SelectionMade += (selectedValue) =>

{

dataGridViewMain.EndEdit();

dataGridViewMain.Rows[e.RowIndex].Cells[e.ColumnIndex].Value = selectedValue;

};

dLinkerForm.ShowDialog();

}

else

{

MessageBox.Show($"Связанная таблица '{tableName}' не найдена.");

}

}

}

else

{

MessageBox.Show("Связанная таблица не найдена.");

}

}

}

}

private void dataGridViewMain\_DataError(object sender, DataGridViewDataErrorEventArgs e)

{

}

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

{

await СоздатьШаблонAsync();

}

private async Task СоздатьШаблонAsync()

{

try

{

ExcelPackage.LicenseContext = LicenseContext.NonCommercial;

using (ExcelPackage excelPackage = new ExcelPackage())

{

ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Читатель");

string[] headers = { "Фамилия", "Имя", "Отчество", "Дата рождения", "Контактный номер",

"Адрес проживания", "Данные паспорта", "Номер читательского билета",

"Название группы", "Год поступления", "Год окончания", "Имя для входа" };

for (int i = 0; i < headers.Length; i++)

{

worksheet.Cells[1, i + 1].Value = headers[i];

}

DataTable groupData = await DBWorker.GetDataTable("Группа");

if (groupData != null && groupData.Rows.Count > 0)

{

ExcelWorksheet groupWorksheet = excelPackage.Workbook.Worksheets.Add("Группы");

int columnCount = 1;

for (int i = 0; i < groupData.Columns.Count; i++)

{

if (!groupData.Columns[i].ColumnName.StartsWith("код", StringComparison.OrdinalIgnoreCase) &&

!groupData.Columns[i].ColumnName.StartsWith("id", StringComparison.OrdinalIgnoreCase))

{

groupWorksheet.Cells[1, columnCount].Value = groupData.Columns[i].ColumnName;

columnCount++;

}

}

for (int row = 0; row < groupData.Rows.Count; row++)

{

columnCount = 1;

for (int col = 0; col < groupData.Columns.Count; col++)

{

if (!groupData.Columns[col].ColumnName.StartsWith("код", StringComparison.OrdinalIgnoreCase) &&

!groupData.Columns[col].ColumnName.StartsWith("id", StringComparison.OrdinalIgnoreCase))

{

groupWorksheet.Cells[row + 2, columnCount].Value = groupData.Rows[row][col];

columnCount++;

}

}

}

}

using (SaveFileDialog saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Filter = "Excel Files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveFileDialog.FileName = "Шаблон\_Читатель.xlsx";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

FileInfo excelFile = new FileInfo(saveFileDialog.FileName);

excelPackage.SaveAs(excelFile);

MessageBox.Show("Шаблон успешно создан.");

}

else {

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при создании шаблона: {ex.Message}");

}

}

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

{

DataTable dt = fetchDataFromExcelFile();

await mergeReadersToDatabase(dt);

}

private DataTable fetchDataFromExcelFile()

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Excel Files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

string filePath = openFileDialog.FileName;

try

{

DataTable excelData = new DataTable();

ExcelPackage.LicenseContext = LicenseContext.NonCommercial;

using (ExcelPackage excelPackage = new ExcelPackage(new FileInfo(filePath)))

{

ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.FirstOrDefault();

if (worksheet != null)

{

for (int col = 1; col <= worksheet.Dimension.End.Column; col++)

{

excelData.Columns.Add(worksheet.Cells[1, col].Value?.ToString());

}

for (int rowNum = 2; rowNum <= worksheet.Dimension.End.Row; rowNum++)

{

DataRow row = excelData.Rows.Add();

for (int col = 1; col <= worksheet.Dimension.End.Column; col++)

{

row[col - 1] = worksheet.Cells[rowNum, col].Value?.ToString();

}

}

return excelData;

}

else

{

throw new Exception("Лист Excel не найден.");

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при обработке данных из файла Excel: {ex.Message}");

}

}

return null;

}

public async Task mergeReadersToDatabase(DataTable dataTable)

{

int groupCode = 0;

if (dataTable is not null)

{

foreach (DataRow row in dataTable.Rows)

{

string groupName = row["Название группы"].ToString();

int yearOfEnrollment = int.Parse(row["Год поступления"].ToString());

int yearOfGraduation = int.Parse(row["Год окончания"].ToString());

int existingGroupCode = await GetGroupCodeIfExists(groupName, yearOfEnrollment, yearOfGraduation);

if (existingGroupCode > 0)

{

groupCode = existingGroupCode;

break;

}

else

{

groupCode = await AddNewGroup(groupName, yearOfEnrollment, yearOfGraduation);

break;

}

}

if (groupCode > 0)

{

await InsertReaders(dataTable, groupCode);

}

}

}

private async Task<int> GetGroupCodeIfExists(string groupName, int yearOfEnrollment, int yearOfGraduation)

{

int groupCode = 0;

using (SqlConnection connection = await ConnectionManager.Instance.OpenConnection())

{

string query = "SELECT Код FROM Группа WHERE Название = @GroupName AND [Год поступления] = @YearOfEnrollment AND [Год окончания] = @YearOfGraduation";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@GroupName", groupName);

command.Parameters.AddWithValue("@YearOfEnrollment", yearOfEnrollment);

command.Parameters.AddWithValue("@YearOfGraduation", yearOfGraduation);

object result = await command.ExecuteScalarAsync();

if (result != null)

{

groupCode = Convert.ToInt32(result);

}

}

return groupCode;

}

private async Task<int> AddNewGroup(string groupName, int yearOfEnrollment, int yearOfGraduation)

{

int groupCode = 0;

using (SqlConnection connection = await ConnectionManager.Instance.OpenConnection())

{

string query = "INSERT INTO Группа (Название, [Год поступления], [Год окончания]) VALUES (@GroupName, @YearOfEnrollment, @YearOfGraduation); SELECT SCOPE\_IDENTITY();";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@GroupName", groupName);

command.Parameters.AddWithValue("@YearOfEnrollment", yearOfEnrollment);

command.Parameters.AddWithValue("@YearOfGraduation", yearOfGraduation);

object result = await command.ExecuteScalarAsync();

if (result != null)

{

groupCode = Convert.ToInt32(result);

}

}

return groupCode;

}

private async Task InsertReaders(DataTable dataTable, int groupCode)

{

using (SqlConnection connection = await ConnectionManager.Instance.OpenConnection())

{

foreach (DataRow row in dataTable.Rows)

{

string surname = row["Фамилия"].ToString();

string name = row["Имя"].ToString();

string patronymic = row["Отчество"].ToString();

DateTime dateOfBirth = DateTime.Parse(row["Дата рождения"].ToString());

string contactNumber = row["Контактный номер"].ToString();

string address = row["Адрес проживания"].ToString();

string passportData = row["Данные паспорта"].ToString();

string readerCardNumber = row["Номер читательского билета"].ToString();

string loginName = row["Имя для входа"].ToString();

string query = "INSERT INTO Читатель ([Фамилия], [Имя], [Отчество], [Дата рождения], [Контактный номер], [Адрес проживания], [Данные паспорта], [Номер читательского билета], [Код группы], [Имя для входа]) " +

"VALUES (@Surname, @Name, @Patronymic, @DateOfBirth, @ContactNumber, @Address, @PassportData, @ReaderCardNumber, @GroupCode, @LoginName)";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Surname", surname);

command.Parameters.AddWithValue("@Name", name);

command.Parameters.AddWithValue("@Patronymic", patronymic);

command.Parameters.AddWithValue("@DateOfBirth", dateOfBirth);

command.Parameters.AddWithValue("@ContactNumber", contactNumber);

command.Parameters.AddWithValue("@Address", address);

command.Parameters.AddWithValue("@PassportData", passportData);

command.Parameters.AddWithValue("@ReaderCardNumber", readerCardNumber);

command.Parameters.AddWithValue("@GroupCode", groupCode);

command.Parameters.AddWithValue("@LoginName", loginName);

await command.ExecuteNonQueryAsync();

}

}

}

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

{

GiveBook b = new();

b.Show();

}

}

}

# ПРИЛОЖЕНИЕ В Код C#, выполняющий логику регистрации и авторизации

namespace LibA

{

public enum WindowType

{

REGISTER,

AUTHORIZE

}

public partial class AuthForm : Form

{

public event EventHandler RegAuthSuccess;

public event EventHandler HasRights;

//private ProgressBar progressBar;

public AuthForm(UserPanel userPanel)

{

InitializeComponent();

this.AutoSizeMode = AutoSizeMode.GrowAndShrink;

this.AutoSize = true;

}

public void WhichWindow(WindowType winType)

{

this.AutoSize = true;

if (winType == WindowType.REGISTER)

{

this.AcceptButton = button1;

authbox.Visible = false;

regbox.Visible = true;

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

this.Text = "Окно регистрации";

}

else if (winType == WindowType.AUTHORIZE)

{

this.AcceptButton = button2;

regbox.Visible = false;

authbox.Visible = true;

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

this.Text = "Окно авторизации";

}

}

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

{

try

{

string transfer = textBox1.Text;

transfer = transfer.Replace(' ', '\_').TrimEnd('\_');

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

//AddProgressBarToRegBox();

//await Task.Delay(10000);

string response = await ConnectionManager.Instance.SendRegData(transfer, textBox2.Text, textBox5.Text);

if (response == "200")

{

ConnectionManager.Instance.SetupConnectionString(textBox2.Text, textBox5.Text);

await ConnectionManager.Instance.OpenConnection();

RegAuthSuccess?.Invoke(this, EventArgs.Empty);

}

else

{

throw new Exception($"{response.Split('|')[1]}");

}

}

catch (SqlException er)

{

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

}

catch (Exception er)

{

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

}

}

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

{

WhichWindow(WindowType.AUTHORIZE);

}

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

{

try

{

ConnectionManager.Instance.SetupConnectionString(textBox3.Text, textBox4.Text);

using (SqlConnection connection = await ConnectionManager.Instance.OpenConnection())

{

if (connection == null)

throw new Exception();

if (connection.State is ConnectionState.Open)

{

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

}

else throw new Exception();

if (await DBWorker.CheckUserRights(await ConnectionManager.Instance.OpenConnection()))

{

HasRights?.Invoke(this, EventArgs.Empty);

}

RegAuthSuccess?.Invoke(this, EventArgs.Empty);

}

this.Close();

}

catch (Exception ex)

{

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

//textBox3.Text = "";

textBox4.Text = "";

textBox3.Focus();

}

}

}

}

# ПРИЛОЖЕНИЕ В Код C#, выполняющий регистрацию пользователя на стороне сервера

namespace SqlCreateUser

{

public partial class SqlServerUserRegister : ServiceBase

{

private TcpListener tcpListener;

private CancellationTokenSource cancellationTokenSource;

private const string CRYPTOKEY = "ThisIsASecretKey1234567890123456";

public SqlServerUserRegister()

{

InitializeComponent();

}

protected override void OnStart(string[] args)

{

cancellationTokenSource = new CancellationTokenSource();

tcpListener = new TcpListener(IPAddress.Any, 8888);

tcpListener.Start();

\_ = AcceptConnectionsAsync(cancellationTokenSource.Token);

}

protected override void OnStop()

{

cancellationTokenSource?.Cancel();

tcpListener?.Stop();

}

private async Task AcceptConnectionsAsync(CancellationToken cancellationToken)

{

try

{

while (!cancellationToken.IsCancellationRequested)

{

TcpClient tcpClient = await tcpListener.AcceptTcpClientAsync();

\_ = HandleClientAsync(tcpClient, cancellationToken);

}

}

catch (Exception ex)

{

// Обработка ошибок при остановке службы

Console.WriteLine(ex.Message);

}

}

private async Task HandleClientAsync(TcpClient tcpClient, CancellationToken cancellationToken)

{

string responce = null;

try

{

NetworkStream networkStream = tcpClient.GetStream();

byte[] data = new byte[1024];

int bytesRead = await networkStream.ReadAsync(data, 0, data.Length, cancellationToken);

string receivedData = Encoding.UTF8.GetString(data, 0, bytesRead);

string[] credentials = receivedData.Split(',');

if (credentials.Length == 3)

{

string userName = credentials[0].Trim();

string login = DecryptString(credentials[1].Trim());

string password = DecryptString(credentials[2].Trim());

string connectionString = $"Data Source=localhost;Initial Catalog=Библиотека;Integrated Security=True";

string query = $"CREATE LOGIN {login} WITH PASSWORD = '{password}'; " +

$"CREATE USER {userName} FOR LOGIN {login}; " +

$"GRANT EXECUTE ON [SearchBooks] TO {userName}";

try

{

using (SqlConnection connectionTRUSTABLE = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connectionTRUSTABLE);

connectionTRUSTABLE.Open();

await command.ExecuteNonQueryAsync(cancellationToken);

string[] ui = userName.Split('\_');

command.CommandText = "INSERT INTO Читатель ([Фамилия], [Имя], [Отчество], [Имя для входа]) VALUES (@Фамилия, @Имя, @Отчество, @ИмяДляВхода)";

command.Parameters.Clear();

command.Parameters.AddWithValue("@Фамилия", ui[0]);

command.Parameters.AddWithValue("@Имя", ui[1]);

command.Parameters.AddWithValue("@Отчество", ui[2]);

command.Parameters.AddWithValue("@ИмяДляВхода", login);

await command.ExecuteNonQueryAsync(cancellationToken);

}

}

catch (Exception e)

{

EventLog.WriteEntry("SQLSlujba", "Ошибка в создании записи.\n" + e.Message, EventLogEntryType.Error);

responce = $"500|{e.Message}";

}

if (responce is null)

responce = "200";

EventLog.WriteEntry("SQLSlujba", "Ответ сервера:\n" + responce, EventLogEntryType.Warning);

byte[] responseData = Encoding.UTF8.GetBytes(responce);

await networkStream.WriteAsync(responseData, 0, responseData.Length);

await networkStream.FlushAsync();

networkStream.Close();

tcpClient.Close();

}

}

catch (Exception ex)

{

Console.WriteLine("finally:" + ex.Message);

}

}

public static string EncryptString(string plainText)

{

using (Aes aesAlg = Aes.Create())

{

aesAlg.Key = Encoding.UTF8.GetBytes(CRYPTOKEY);

aesAlg.IV = new byte[aesAlg.BlockSize / 8];

ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);

using (MemoryStream msEncrypt = new MemoryStream())

{

using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))

using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))

{

swEncrypt.Write(plainText);

}

return Convert.ToBase64String(msEncrypt.ToArray());

}

}

}

public static string DecryptString(string cipherText)

{

using (Aes aesAlg = Aes.Create())

{

aesAlg.Key = Encoding.UTF8.GetBytes(CRYPTOKEY);

aesAlg.IV = new byte[aesAlg.BlockSize / 8];

ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);

using (MemoryStream msDecrypt = new MemoryStream(Convert.FromBase64String(cipherText)))

{

using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))

using (StreamReader srDecrypt = new StreamReader(csDecrypt))

{

return srDecrypt.ReadToEnd();

}

}

}

}

}

# ПРИЛОЖЕНИЕ Д Результат проверки на плагиат