**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КОМИ**

**Государственное профессиональное образовательное учреждение**

**«Воркутинский политехнический техникум»**

**КУРСОВОЙ ПРОЕКТ**

По дисциплине МДК.05.02 Разработка кода

информационных систем

Разработка информационной системы «Библиотека»

Выполнил студент гр. ИСП-20 /\_\_\_\_\_\_\_\_\_\_\_\_\_/Зверева.А.И. /

(подпись) (Ф.И.О.)

**ОЦЕНКА: \_\_\_\_\_\_\_\_\_\_\_\_\_**

Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРОВЕРИЛ**

Научный руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_/ Егоров Данил Павлович/

(подпись) (Ф.И.О.)

Воркута

2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc133403045)

[ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 4](#_Toc133403046)

[1.1 Платформа .NET 4](#_Toc133403047)

[1.2 Язык программирования С# 4](#_Toc133403048)

[1.3 Windows Presentation Foundation (WPF) 5](#_Toc133403049)

[1.4 СУБД SQL server 7](#_Toc133403050)

[1.5 Microsoft SQL Server Management Studio 8](#_Toc133403051)

[1.6 Entity Framework 9](#_Toc133403052)

[ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ 11](#_Toc133403053)

[2.1 Разработка диаграммы ERD 11](#_Toc133403054)

[2.2 Разработка базы данных 12](#_Toc133403055)

[ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 16](#_Toc133403056)

[3.1 Разработка интерфейса информационной системы 16](#_Toc133403057)

[3.2 Программирование информационной системы 24](#_Toc133403058)

[ЗАКЛЮЧЕНИЕ 30](#_Toc133403059)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 31](#_Toc133403060)

ВВЕДЕНИЕ

Библиотека — учреждение, собирающее и осуществляющее хранение произведений печати и письменности для общественного пользования. Деятельность библиотек по обслуживанию читателей осуществляется в двух основных формах. Библиотечный абонемент предоставляет читателю право получить издание из библиотеки в своё полное распоряжение на определённый срок. В другом случае читатель имеет возможность знакомиться с книгой только в помещении библиотеки.

Развитие библиотечной деятельности неразделимо с внедрением передовых технологий и процессов с целью обеспечения наиболее качественного обслуживания читателей. Внедрение автоматизированных информационных систем – это не только тенденция времени, но и хорошая база для увеличения производительности и качества труда сотрудников библиотеки, эффективный способ своевременного обеспечения читателей необходимыми им данными и информацией.

С помощью запросов реализованы функции поиска информации по заданной тематике, группировки информации для различных отчетов, редактирования и удаления записей в таблицах. Поиск книг, статей по автору, названию или соответствию некоторому разделу осуществляется с помощью запросов с параметром. Их удобно использовать в качестве источников записей для форм, отчетов и страниц доступа к данным.

**Объект** - информационная система «Библиотека».

**Предмет** - автоматизация бизнес процессов учета книг «Библиотека»

**Цель:** создание информационной системы «Библиотека»

**Задачи:**

* выбрать инструментарий;
* разработать баз данных;
* разработать информационную систему.

ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

* 1. Платформа .NET

Платформа .NET Framework состоит из общеязыковой среды выполнения (среды CLR) и библиотеки классов .NET Framework. Основой платформы .NET Framework является среда CLR. Среду выполнения можно считать агентом, который управляет кодом во время выполнения и предоставляет основные службы, такие как управление памятью, управление потоками и удаленное взаимодействие. При этом средой накладываются условия строгой типизации и другие виды проверки точности кода, обеспечивающие безопасность и надежность. Фактически основной задачей среды выполнения является управление кодом. Код, который обращается к среде выполнения, называют управляемым кодом, а код, который не обращается к среде выполнения, называют неуправляемым кодом. Библиотека классов является комплексной объектно-ориентированной коллекцией повторно используемых типов, которые применяются для разработки приложений — начиная с обычных приложений, запускаемых из командной строки, и приложений с графическим интерфейсом (GUI) и заканчивая приложениями, использующими последние технологические возможности ASP.NET, такие как веб-формы и веб-службы XML.

Платформа .NET Framework может размещаться неуправляемыми компонентами, которые загружают среду CLR в собственные процессы и запускают выполнение управляемого кода, создавая таким образом программную среду, позволяющую использовать средства как управляемого, так и неуправляемого выполнения. Платформа .NET Framework не только предоставляет несколько базовых сред выполнения, но также поддерживает разработку базовых сред выполнения независимыми производителями.

* 1. Язык программирования С#

C# — объектно-ориентированный, ориентированный на компоненты язык программирования. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО. В основном C# — объектно-ориентированный язык. Вы определяете типы и их поведение.

Вот лишь несколько функций языка C#, которые позволяют создавать надежные и устойчивые приложения. Сборка мусора автоматически освобождает память, занятую недостижимыми неиспользуемыми объектами. Типы, допускающие значение null, обеспечивают защиту от переменных, которые не ссылаются на выделенные объекты. Обработка исключений предоставляет структурированный и расширяемый подход к обнаружению ошибок и восстановлению после них. Лямбда-выражения поддерживают приемы функционального программирования. Синтаксис LINQ создает общий шаблон для работы с данными из любого источника. Поддержка языков для асинхронных операций предоставляет синтаксис для создания распределенных систем. В C# имеется Единая система типов. Все типы C#, включая типы-примитивы, такие как int и double, наследуют от одного корневого типа object. Все типы используют общий набор операций, а значения любого типа можно хранить, передавать и обрабатывать схожим образом. Более того, C# поддерживает как определяемые пользователями ссылочные типы, так и типы значений. C# позволяет динамически выделять объекты и хранить упрощенные структуры в стеке. C# поддерживает универсальные методы и типы, обеспечивающие повышенную безопасность типов и производительность. C# предоставляет итераторы, которые позволяют разработчикам классов коллекций определять пользовательские варианты поведения для клиентского кода.

C# подчеркивает управление версиями , чтобы обеспечить совместимость программ и библиотек с течением времени. Вопросы управления версиями существенно повлияли на такие аспекты разработки C#, как раздельные модификаторы virtual и override, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса.

1.3 Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) представляет собой обширный API-интерфейс для создания настольных графических программ имеющих насыщенный дизайн и интерактивность. В отличие от устаревшей технологии Windows Forms, WPF включает новую модель построения пользовательских приложений (в основе WPF лежит мощная инфраструктура, основанная на DirectX).

Это означает возможность применения развитых графических эффектов, не платя за это производительностью, как это было в Windows Forms. Фактически даже становятся доступными такие расширенные средства, как поддержка видеофайлов и трехмерное содержимое. Используя эти средства (при наличии хорошего инструмента графического дизайна), можно создавать бросающиеся в глаза пользовательские интерфейсы и визуальные эффекты, которые были просто невозможны в Windows Forms.

Элементы управления чаще всего используются для определения ввода данных пользователем и реагирования на него. Система ввода WPF использует как прямые, так и перенаправленные события для поддержки ввода текста, управления фокусом и определения положения указателя мыши.

Приложения часто предъявляют сложные требования к вводу. WPF предоставляет систему команд, которая отделяет действия по вводу данных пользователем от кода, реагирующего на эти действия. Система команд позволяет нескольким источникам вызывать одну и ту же логику команд.

WPF представляет обширный, масштабируемый и гибкий набор графических возможностей:

Графика, не зависящая от разрешения и устройства. Основной единицей измерения в графической системе WPF является аппаратно-независимый пиксель, который составляет 1/96 часть дюйма независимо от фактического разрешения экрана.

Дополнительная поддержка графики и анимации. WPF упрощает программирование графики за счет автоматического управления анимацией. Разработчик не должен заниматься обработкой сцен анимации, циклами отрисовки и билинейной интерполяцией.

Web-подобная модель компоновки. Вместо того чтобы фиксировать элементы управления на месте с определенными координатами, WPF поддерживает гибкий поток, размещающий элементы управления на основе их содержимого. В результате получается пользовательский интерфейс, который может быть адаптирован для отображения высокодинамичного содержимого или разных языков;

• Богатая модель рисования. Вместо рисования пикселей в WPF имеет дело с примитивами – базовыми фигурами, блоками текста и другими графическими элементами. Также имеются такие новые средства, как действительно прозрачные элементы управления, возможность складывать множество уровней с разной степенью прозрачности, а также встроенную поддержку трехмерной графики;

• Богатая текстовая модель. После многих лет нестандартной обработки текстов в таких несовершенных элементах управления, как классический Label,WPF предоставляет приложениям Windows возможность отображения богатого стилизованного текста в любом месте пользовательского интерфейса. И если необходимо отображать значительные объемы текста, можно воспользоваться развитыми средствами отображения документов, такими как переносы, разбиение на колонки и выравнивание для повышения читабельности;

• Анимация как программная концепция. В WPF анимация – неотъемлемая часть программного каркаса. Она определяется декларативными дескрипторами, и WPF запускает ее в действие автоматически;

Элементы управления являются основой любого графического интерфейса. WPF предлагает как уже знакомые по Windows Forms элементы управления (Button, CheckBox и т.д.), так и совершенно новые (Expander, Page и т.д.). В данном разделе вы найдете также описание довольно распространенного элемента Ribbon, не входящего в состав .NET Framework.

1.4 СУБД SQL server

СУБД SQL server используются для создания, размещения, хранения и управления реляционными (табличными) базами данных на специальных серверах или в облаке. Они работают через настольные приложения и web-сайты. К основным преимуществам их функционирования относятся:

* высокоскоростной доступ к данным, обеспечиваемый надежной клиент-серверной архитектурой СУБД;
* простота работы и администрирования, обусловленные понятной структурой языка программирования SQL;
* безопасность хранения информации в БД - благодаря возможности шифрования данных и резервного копирования.
* Специфика работы сервера базы данных SQL server заключается в транзакционной обработке данных. Это означает, что по каждому запросу от СУБД обрабатывается и сохраняется небольшое количество информации.
* Применение SQL server позволяет автоматизировать решение различных бизнес-задач, поддерживать проведение аналитики данных в режиме онлайн, отслеживать направление ресурсов СУБД, управлять транзакциями (операциями по обработке данных).

СУБД служит посредником между пользователем и БД. Сама структура БД хранится в виде набора файлов, и единственный способ получить доступ к данным в этих файлах – через СУБД. Она представляет конечному пользователю (или прикладной программе) единое интегрированное представление данных в БД. СУБД получает запросы приложений и переводит их в сложные операции, необходимые для выполнения. Она скрывает большую часть внутренней сложности БД от прикладных программ и пользователей. Прикладная программа может быть написана программистом с использованием языка программирования высокого уровня, такого как Python, Java или C#, или она может быть создана с помощью инструментов СУБД. Наличие СУБД между приложениями пользователя и БД дает некоторые важные преимущества. Во-первых, СУБД позволяет совместно использовать данные в БД нескольким приложениям или пользователям. Во-вторых, СУБД объединяет представления данных разных пользователей в единый универсальный репозиторий данных.

Большинство систем баз данных содержат подкомпонент, называющийся оптимизатором, который рассматривает несколько возможных стратегий исполнения запроса данных и выбирает из них наиболее эффективную. Выбранная стратегия называется планом исполнения запроса. Оптимизатор принимает решение, принимая во внимание такие факторы, как размер таблиц, к которым направлен запрос, существующие индексы и логические операторы (AND, OR или NOT), используемые в предложении WHERE.

Одной из стоящих перед системой баз данных задач является идентифицировать логически противоречивые данные и не допустить их помещения в базу данных. Кроме этого, для большинства реальных задач, которые реализовываются с помощью систем баз данных, существуют ограничения для обеспечения целостности (integrity constraints), которые должны выполняться для данных. (В качестве примера ограничения для обеспечения целостности можно назвать требование, чтобы табельный номер сотрудника был пятизначным целым числом.)

Обеспечение целостности данных может осуществляться пользователем в прикладной программе или же системой управления базами данных. До максимально возможной степени эта задача должна осуществляться посредством СУБД.

1.5 Microsoft SQL Server Management Studio

Среда SQL Server Management Studio предназначена для доступа к службам Службы Analysis Services, Integration Services и Reporting Services, а также для их настройки, администрирования и управления ими. Хотя все три технологии бизнес-аналитики полагаются на среду SQL Server Management Studio, административные задачи, связанные с каждой из этих технологий, несколько отличаются.

Среда SQL Server Management Studio позволяет управлять объектами Службы Analysis Services, например выполнять их резервное копирование и обработку.

Среда Management Studio позволяет создавать проекты скриптов служб Службы Analysis Services, в которых выполняются разработка и сохранение скриптов с использованием многомерных выражений (MDX), расширений интеллектуального анализа данных (DMX) и XML для аналитики (XMLA). Проекты скриптов служб Службы Analysis Services используются для выполнения задач управления или повторного создания баз данных, кубов и других объектов в экземплярах служб Службы Analysis Services. Например, можно разработать скрипт XMLA в проекте скрипта служб Службы Analysis Services , который создает объекты непосредственно в существующем экземпляре служб Службы Analysis Services . Проекты скриптов служб Службы Analysis Services могут быть сохранены в составе решения и интегрироваться с контролем исходного кода.

1.6 Entity Framework

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

Первая версия Entity Framework - 1.0 вышла еще в 2008 году и представляла очень ограниченную функциональность, базовую поддержку ORM (object-relational mapping - отображения данных на реальные объекты) и один единственный подход к взаимодействию с бд - Database First. С выходом версии 4.0 в 2010 году многое изменилось - с этого времени Entity Framework стал рекомендуемой технологией для доступа к данным, а в сам фреймворк были введены новые возможности взаимодействия с бд - подходы Model First и Code First.

Дополнительные улучшения функционала последовали с выходом версии 5.0 в 2012 году. И наконец, в 2013 году был выпущен Entity Framework 6.0, обладающий возможностью асинхронного доступа к данным.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст, вес. Свойства необязательно представляют простые данные типа int, но и могут представлять более комплексные структуры данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять эту сущность. Подобные свойства называют ключами.

При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из бд, но и получать объекты, связанные различными ассоциативными связями.

ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ

2.1 Разработка диаграммы ERD

Диаграммы отношений сущностей (ER) - это визуальные представления баз данных, которые показывают, как элементы в базе данных связаны друг с другом. Примерами сущностей (с точки зрения логической модели предметной области) или объектов (с точки зрения внешнего мира по отношению к ИС). ER-диаграммы — «родственники» схем структуры данных (DSD), где вместо связей между самими сущностями отображаются отношения между элементами внутри них. ER-диаграммы часто используются в сочетании с диаграммами DFD, которые схематично показывают движение потоков информации в рамках процесса или системы. С объектами связано две проблемы — идентификация и адекватное описание. Для идентификации используют имя. При этом предполагается, что происходит отказ от его смысла, который присущ естественному языку. Используется только указательная функция имени. Имя — это прямой способ идентификации объекта. К косвенным способам идентификации объекта относят его свойства в их понимании как характеристики или признака.

Объекты взаимодействуют между собой через свои свойства, что порождает ситуации. Ситуации – это взаимосвязи, выражающие взаимоотношения между объектами. Ситуации в предметной области описываются посредством высказываний о предметной области с использованием исчисления высказываний и исчисления предикатов, т.е. формальной, математической логики.

ER-диаграммы применяются для моделирования и проектирования реляционных баз данных, причем как в плане логических и бизнес-правил (логические модели данных), так и в плане внедрения конкретных технологий (физические модели данных). В сфере разработки программного обеспечения ER-диаграмма, как правило, служит первым шагом в определении требований проекта по созданию информационных систем. На дальнейших этапах работы ER-диаграммы также применяются для моделирования конкретных баз данных. Реляционная база данных сопровождается соответствующей реляционной таблицей и при необходимости может быть представлена в этом формате.

Диаграммы «сущность-связь» (или ERD) — неотъемлемая составляющая процесса моделирования любых систем, включая простые и сложные базы данных, однако применяемые в них фигуры и способы нотации могут запросто ввести в заблуждение любого.

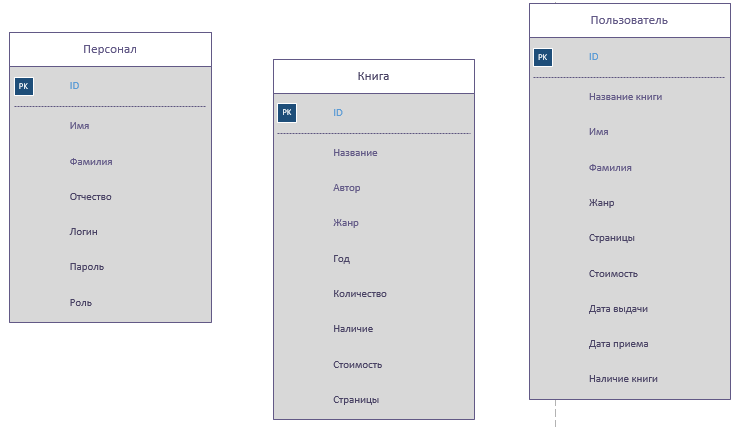


Рисунок 2.1. ER

Основными сущностями являются: персонал, книга, пользователь. Каждая сущность обделена следующими атрибутами. У сущности «Персонал» имеется: ФИО, логин, пароль, роль. У сущности «Книга» имеется: название, автор, жанр, год, количество, наличие, стоимость, страницы. У сущности «Пользователь» имеется: название книги, имя, фамилия, жанр, страницы, стоимость, дата выдачи, дата приема, наличие книги.

2.2 Разработка базы данных

Затем следует процесс разработки базы данных на языке SQL. Средствами для создания БД являются БД SQL Server Management Studio.

База данных АИС «Библиотека» была подготовлена способом создания диаграммы. Диаграммы базы данных графически изображают структуру баз данных. При помощи диаграмм баз данных можно создавать и изменять таблицы, столбцы, связи и ключи. Кроме того, можно изменять индексы и ограничения. Таблицы являются объектами, которые содержат все данные в базах данных. В таблицах данные логически организованы в виде строк и столбцов по аналогии с электронной таблицей. Каждая строка представляет собой уникальную запись, а каждый столбец — поле записи. Например, таблица, содержащая данные о сотрудниках компании, может иметь строку для каждого сотрудника и столбцы, представляющие сведения о сотрудниках (например, его идентификационный номер, имя, адрес и т.д.).

Итогом реализации АИС «Библиотека» предоставляется данная база данных:

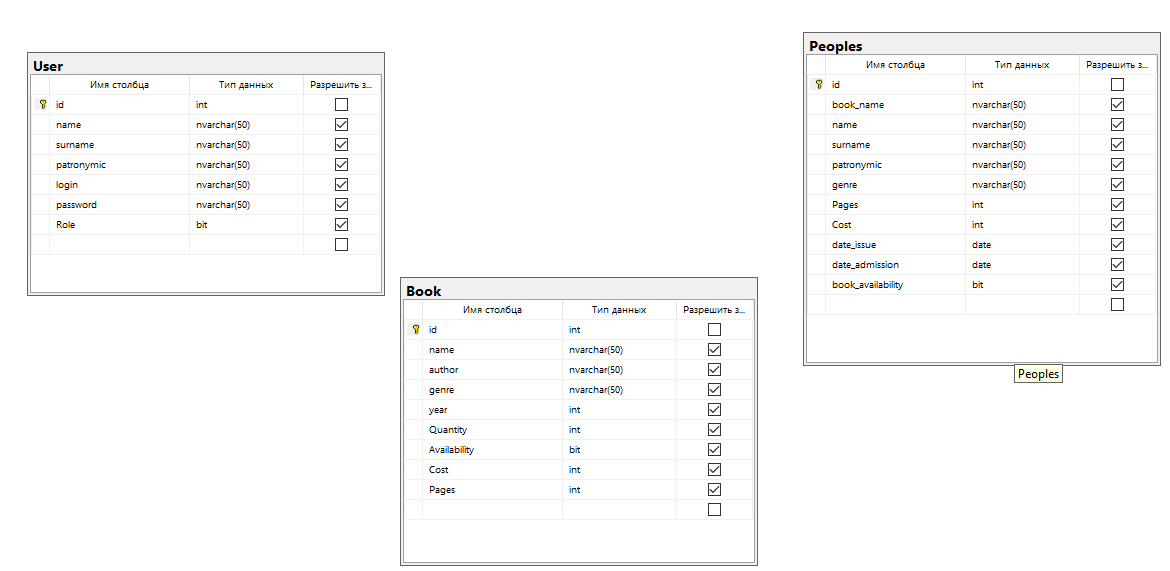
****

Рисунок 2.2. База данных

БД состоит из 3-х таблиц: таблица с отчётами о данных персонала, таблица с отчётами о книгах, таблица пользователей. В каждой таблице содержится свой персональный ключ, который позволяет связываться с другой таблицей, создавая в нём внешний ключ.

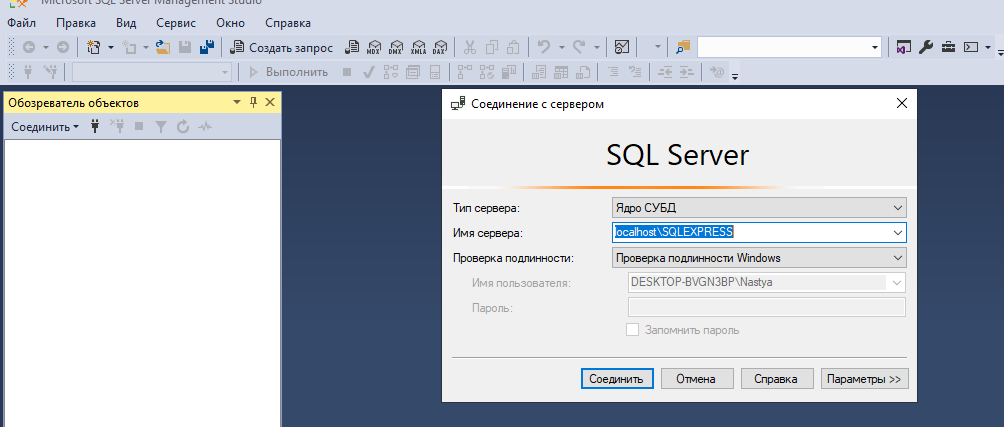


Рисунок 2.3 Подключение к локальному серверу

1. Создание БД, чтобы выполнить данную операцию, нужно нажать ПКМ по “Базы данных”, выбрать пункт “Создать базу данных” и задать имя:

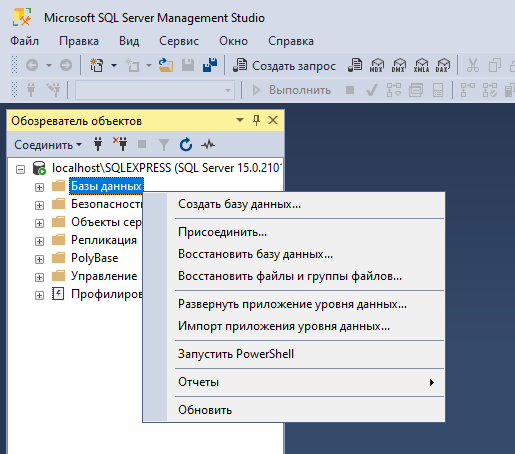


Рисунок 2.4. Создание Базы данных

1. Создание диаграммы, чтобы выполнить данную операцию, нужно нажать ПКМ по “Базы данных” и выбрать “Создать диаграмму базы данных”:

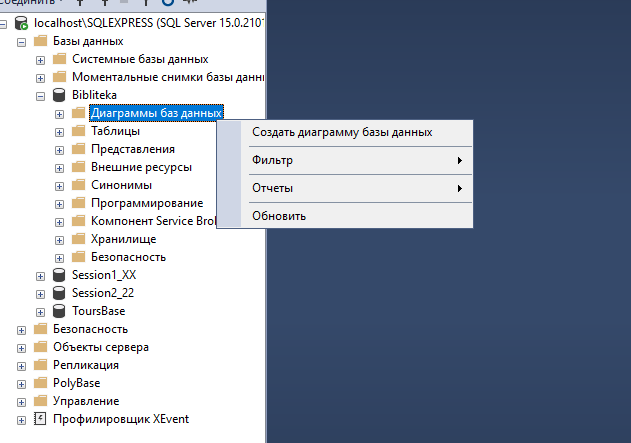


Рисунок 2.5. Создание диаграммы

1. Создание таблиц, чтобы выполнить данную операцию, нужно нажать ПКМ по пустой области и выбрать пункт “Создать таблицу”, ввести имя таблицы и настроить данные о таблице:

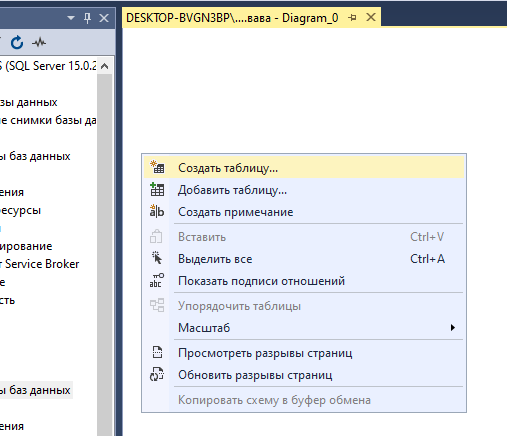


Рисунок 2.6. Создание и настройка таблицы

1. Введение данных в таблицу, чтобы выполнить данную операцию, нужно открыть папку tables, нажать ПКМ по таблице и выбрать пункт “Изменить первые 200 строк”:

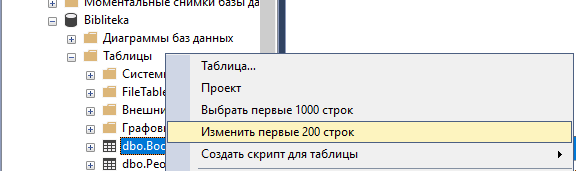


Рисунок 2.7. Таблицы

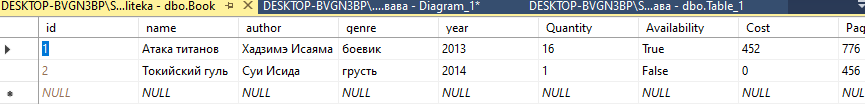
1. Пример одной из таблиц: 

Рисунок 2.8. Пример таблицы

ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1 Разработка интерфейса информационной системы

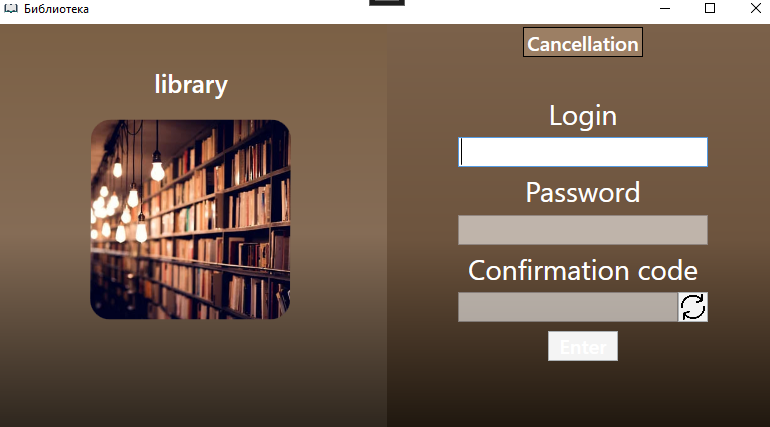


Рисунок 3.1 Форма Авторизации

Автоматизированная информационная система (АИС) состоит из взаимосвязанной совокупности подразделений организации и комплекса средств автоматизации деятельности, и реализует автоматизированные функции по отдельным видам деятельности. Разновидностью АС являются информационные системы (ИС), основной целью которых является хранение, обеспечение эффективного поиска и передачи информации по соответствующим запросам.

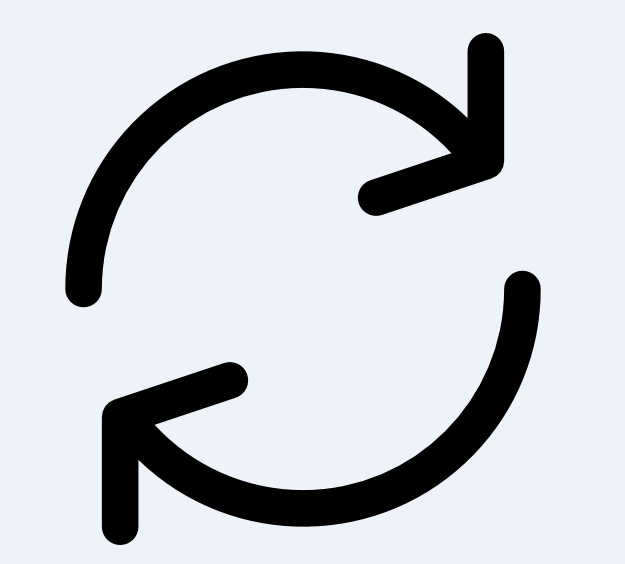
При этом автоматизированные информационные системы (АИС) являются областью информатизации, механизмом и технологией, эффективным средством обработки, хранения, поиска и представления информации потребителю. АИС представляют совокупность функциональных подсистем сбора, ввода, обработки, хранения, поиска и распространения информации.

Автоматизированная информационная система «Библиотека» должна обеспечивать выполнение функций:

* Ввод данных читателей;
* Формирование отчетов по выдаче книг читателю;
* Формирование отчета по книгам;
* Отчет по должникам.

АИС «Библиотека» встречает персонал окном авторизации. На данном этапе реализованы поля для ввода данных персонала: логин и пароль. Также имеется код, который дает возможность рандомно обработать в системе, блок генерации кода, который можно обновить. После авторизации, появится окно с сообщением “Вы успешно вошли!!”

Также в данном окне сверху пользователь может увидеть иконку приложения, а на фоне стоит приятный градиентный тон. На левой стороне интерфейса расположена картинка с библиотекой с закругленными краями. Поля входа сделаны также в приятных цветах. Функционал 3-ех кнопок:

Кнопка “Enter” – отвечает да вход, с помощью нее сотрудник сможет авторизоваться в свою информационную систему. Кнопка “Cancellation” – отвечает за отмену действия выполненным сотрудником.(Сотрудник не правильно заполнил поля логина или пароля, функционал этой кнопки может помочь отменить свои действия). Кнопка  - отвечает за обновление обработки кода.

**Xaml код**:

<Grid.Background>

Код, отвечающий за цвет фона:

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF7B614A"/>

<GradientStop Color="#FF1C150D" Offset="1"/>

<GradientStop Color="#FF6D543E" Offset="0.524"/>

</LinearGradientBrush>

</Grid.Background>

Код, определяющий количество строк в полях:

<Grid.RowDefinitions>

<RowDefinition Height="80"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

Код, который обрабатывает картинку так, чтобы она была с закругленными краями:

<Image Width="200" Height="200" Stretch="UniformToFill"

Source="/Resources/фото.jpg" RenderTransformOrigin="0.5,0.5" Margin="40 0 40 100" Grid.Row="1">

<Image.RenderTransform>

<TransformGroup>

Код, отвечающий за надпись логина:

<TextBlock Text="Login" FontSize="28" Grid.Row="0" TextAlignment="Right" VerticalAlignment="Center" HorizontalAlignment="Center" Foreground="White"/>

Код, отвечающий за поле ввода логина:

<TextBox Name ="Log" KeyUp="Log\_KeyUp" Height="30" Width="250" FontSize="20" FontWeight="DemiBold" Grid.Row="1"></TextBox>

Код, отвечающий за надпись пароля:

<TextBlock Text="Password" FontSize="28" Grid.Row="2" TextAlignment="Right" VerticalAlignment="Center" HorizontalAlignment="Center" Foreground="White"/>

Код, отвечающий за поле ввода пароля:

<PasswordBox Name ="Pass" KeyUp="Pass\_KeyUp" Height="30" Width="250" FontSize="20" FontWeight="DemiBold" Grid.Row="3"></PasswordBox>

Код, отвечающий за поле по отмене действий пользователем:

<TextBlock Text="Сonfirmation code" FontSize="28" Grid.Row="4" TextAlignment="Right" VerticalAlignment="Center" HorizontalAlignment="Center" Foreground="White"/>

Код, отвечающий за расположение полей:

<DockPanel Grid.Row="5" HorizontalAlignment="Center" VerticalAlignment="Center">

Код, отвечающий за надпись кода:

<TextBox Name ="Code" Height="30" Width="220" FontSize="20" FontWeight="DemiBold"></TextBox>

Код, отвечающий за обновление генерации рандомного кода:

<Button Name="Refrsh" Width="30" Height="30" Click="Refrsh\_Click">

<Image Source="/Resources/free-icon-refresh-875100.png" />

</Button>

</DockPanel>

<Image Source="/Resources/free-icon-refresh-875100.png" />

</Button>

</DockPanel>

<Button Name="Enter" Grid.Row="6" Content="Enter" Width="70" Height="30" FontSize="20" FontWeight="DemiBold" VerticalAlignment="Center" HorizontalAlignment="Center" Click="Enter\_Click" Background="#FF9C7F64" Foreground="White"/>

</Grid>

Код, отвечающий на кнопку “Отмена”

<Button x:Name="ClearBut" FontSize="20" Content="Cancellation" FontWeight="DemiBold" Background="#FF9C7F64" BorderBrush="Black" Foreground="White" HorizontalAlignment="Center" Margin="0,3,0,0" VerticalAlignment="Top" Height="30" Width="120" Click="ClearBut\_Click" Grid.Column="1"/>

</Grid>

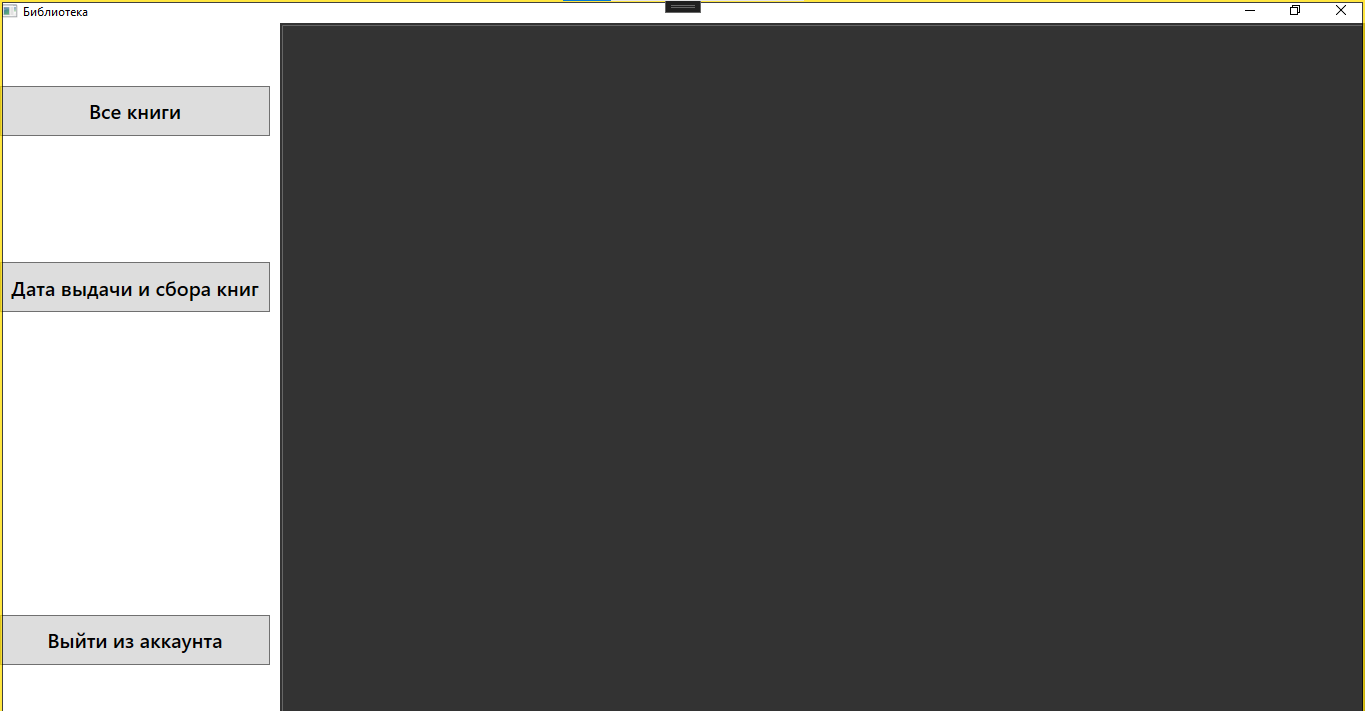


Рисунок 3.2. Форма главного окна

После успешной авторизации, появляется главное окно. В данном окне есть возможность перейти на вкладки: “Все книги”, “Дата выдачи и сбора книг”, и “Выйти из аккаунта”.

После того, как мы нажали кнопку “Все книги”, открывается окно с данными о книгах в библиотеке.

При нажатии на кнопку “Дата выдачи и сбора книг”, открывается окно с индентичными данными, только добавляется отчёты о дате взятия книги и возврате.

При нажатии на кнопку “Выйти из аккаунта”, закрывается главное окно и пользователя обратно переносит на окно авторизации.

Xaml код:

<Grid>

Код, отвечающий за расположение строк:

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

Код, указывающий каком столбце и строке будет находиться элемент:

<Grid Grid.Row="0">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

Код, отвечающий за кнопку “Все книги”:

<Button Name="Books" Click="Books\_Click" Content="Все книги" Grid.Row="0" Grid.Column="0" Margin="0 0 10 0" Height="50" Width="270" FontSize="20" FontWeight="DemiBold" VerticalAlignment="Center" HorizontalAlignment="Left"/>

Код, отечающий за кнопку “Дата и выдачи сбора книг”

<Button Name="AllData" Click="AllData\_Click" Content="Дата выдачи и сбора книг" Grid.Row="1" Margin="0 0 10 0" Grid.Column="0" Height="50" Width="270" FontSize="20" FontWeight="DemiBold" VerticalAlignment="Center" HorizontalAlignment="Left"/>

Код, отвечающий за кнопку “Выйти из аккаунта”:

<Button Name="Exit" Content="Выйти из аккаунта" Click="Exit\_Click" Grid.Row="3" Grid.Column="0" Margin="0 0 10 0" Height="50" Width="270" FontSize="20" FontWeight="DemiBold" VerticalAlignment="Center" HorizontalAlignment="Left"/>

</Grid>

Код, отвечающий за главный фрейм:

<Frame x:Name="MainFrame" Background="Black" Opacity="0.8" BorderBrush="Black" BorderThickness="2.5" NavigationUIVisibility="Hidden" Grid.Column="1"/>

</Grid>

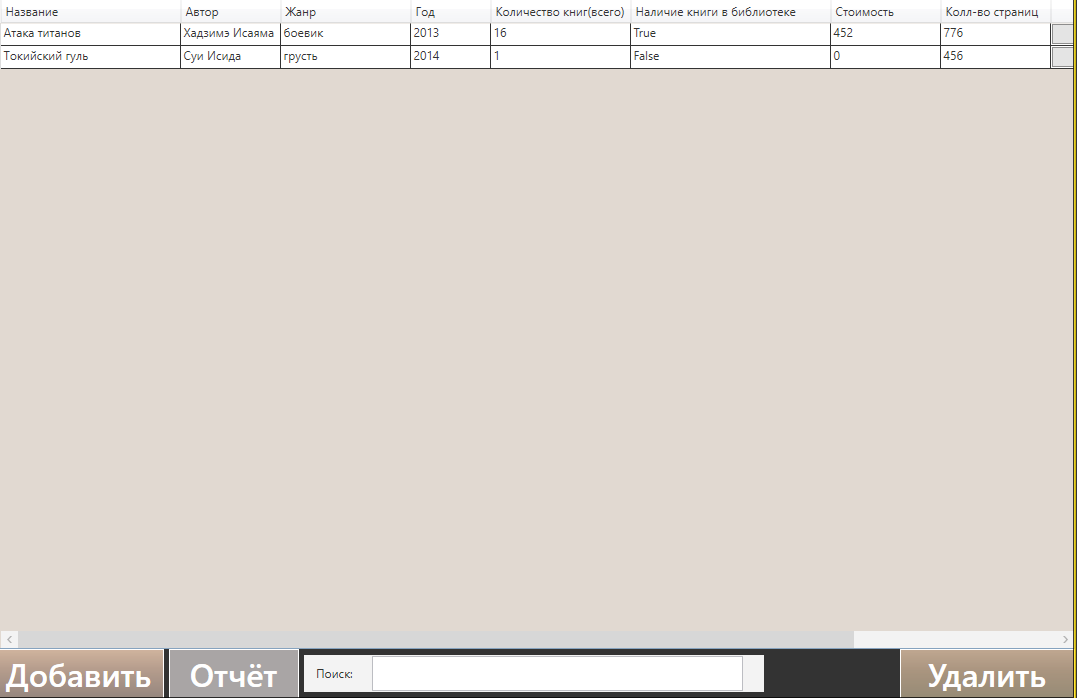


Рисунок 3.3. Окно с данными о книгах

После того как пользователь перешел по кнопке “Книги”, он может посмотреть все данные о книгах. В таблице сформированы такие данные как: название, автор, жанр, год, количество книг(всего), наличие книги в библиотеке, стоимость и количество страниц. Также каждую из ячеек из таблицы предоставляет возможность редактирования.

Помимо всего функционала, есть возможность добавить книгу с помощью нажатия на кнопку “Добавить”.

Также пользователь имеет возможность удалить выбранные данные с помощью нажатия на кнопку “Удалить”. Стоит отметить, что также на данном окне есть поисковая строка, где можно найти информацию из базы данных, а также кнопка “Отчёты”, после нажатия которой, откроется файл Word, где будут выведены все данные в виде таблицы.

Xaml код:

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<DataGrid Name="DGridBooks" AutoGenerateColumns="False" IsReadOnly="True" Background="#FFD9CFC5">

<DataGrid.Columns>

Код, отвечающий за надпись “Название”:

<DataGridTextColumn Header="Название" Binding="{Binding name}" Width="180"/>

Код, отвечающий за надпись ”Автор”:

<DataGridTextColumn Header="Автор" Binding="{Binding author}" Width="100"/>

Код, отвечающий за надпись “Жанр”:

<DataGridTextColumn Header="Жанр" Binding="{Binding genre}" Width="130"/>

Код, отвечающий за надпись ”Год”:

<DataGridTextColumn Header="Год" Binding="{Binding year}" Width="80"/>

Код, отвечающий за надпись “Количество книг(всего)”:

<DataGridTextColumn Header="Количество книг(всего)" Binding="{Binding Quantity}" Width="140"/>

Код, отвечающий за надпись “Наличие книги в библиотеке”:

<DataGridTextColumn Header="Наличие книги в библиотеке" Binding="{Binding Availability}" Width="200"/>

Код, отвечающий за надпись “Стоимость”:

<DataGridTextColumn Header="Стоимость" Binding="{Binding Cost}" Width="110"/>

Код, отвечающий за надпись “Количество страниц”:

<DataGridTextColumn Header="Колл-во страниц" Binding="{Binding Pages}" Width="110"/>

<DataGridTemplateColumn Width="285">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

Код, отвечающий за кнопку”Редактировать”:

<Button Content="Редактировать" Name="BtnEdit" Click="BtnEdit\_Click"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<StackPanel Orientation="Horizontal" Grid.Row="1" HorizontalAlignment="Center" VerticalAlignment="Center">

<Menu Width="460">

Код, отвечающий за поле “Поиск”:

<Label Content="Поиск:"/>

<TextBox Width="371" Height="35" Name="TBoxSearch1" TextChanged="TBoxSearch1TextChanged" FontSize="22" FontWeight="DemiBold"/>

</Menu>

</StackPanel>

<Button Name="BtnAdd" Click="BtnAdd\_Click" Content="Добавить" HorizontalAlignment="Left" Width="170" BorderBrush="White" Foreground="White" FontSize="32" FontWeight="DemiBold" Grid.Row="1">

Код, отвечающий за фон:

<Button.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FFC4A185"/>

<GradientStop Color="#FF7E6A5F" Offset="0.961"/>

<GradientStop Color="#FFA48571" Offset="0.395"/>

</LinearGradientBrush>

</Button.Background>

</Button>

<Button Name="BtnDel" Click="BtnDel\_Click" Content="Удалить" HorizontalAlignment="Left" Width="175" BorderBrush="White" Foreground="White" FontSize="32" FontWeight="DemiBold" Margin="906,0,0,0" Grid.Row="1">

<Button.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FFAF9175"/>

<GradientStop Color="#FF7C6E53" Offset="1"/>

<GradientStop Color="#FF947A5F" Offset="0.442"/>

</LinearGradientBrush>

</Button.Background>

</Button>

Кнопка, отвечающая за экспорт отчетов в Word:

<Button Name="Otchet" Click="Otchet\_Click" Content="Отчёт" HorizontalAlignment="Left" Width="130" BorderBrush="White" Foreground="White" FontSize="32" FontWeight="DemiBold" Background="#FF948E8E" Margin="175,0,0,0" Grid.Row="1"/>

</Grid>

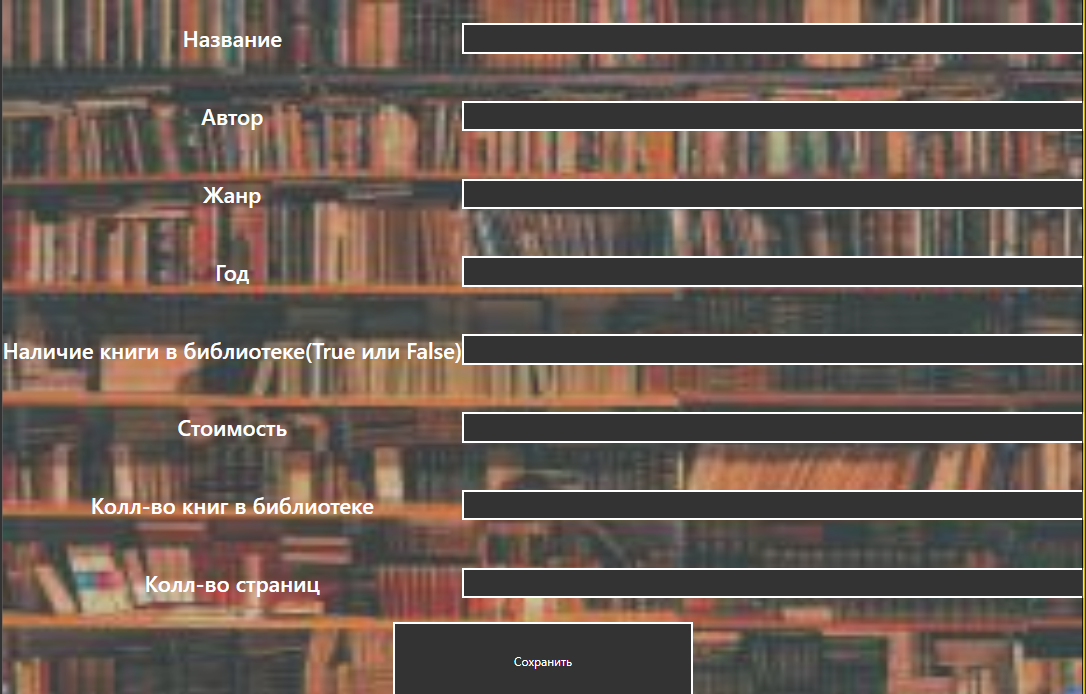


Рисунок 3.4. Добавление

Данное поле используется при добавлении книг. При использовании данного поля необходимо ввести название книги, автора, жанр, год выпуска, наличие книги в библиотеке(1 или 2), стоимость, количество книг в библиотеке, количество страниц.

Xaml код:

</Grid.RowDefinitions>

Код, отвечающий за надпись “Имя”:

<TextBlock Text="Имя" Grid.Row="0" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

Код, отвечающий за надпись “Фамилия”:

<TextBlock Text="Фамилия" Grid.Row="1" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

Код, отвечающий за надпись “Отчество”:

<TextBlock Text="Отчество" Grid.Row="2" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

Код, отвечающий за надпись “Название книги”:

<TextBlock Text="Название книги" Grid.Row="3" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

Код, отвечающий за надпись “Жанр”:

<TextBlock Text="Жанр" Grid.Row="4" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

Код, отвечающий за надпись “Количество страниц”:

<TextBlock Text="Колл-во страниц" Grid.Row="5" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

<TextBlock Text="Стоимость" Grid.Row="6" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

Код, отвечающий за надпись “Дата получения”:

<TextBlock Text="Когда получил" Grid.Row="7" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

Код, отвечающий за надпись “Когда должен был вернуть”:

<TextBlock Text="Когда должен был вернуть" Grid.Row="8" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

Код, отвечающий за надпись “Вернул ли?(1 или 2) ”:

<TextBlock Text="Вернул ли?(True, False)" Grid.Row="9" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

<TextBox Text="{Binding name}" VerticalAlignment="Center" FontSize="20" MaxLength="100" Grid.Column="1" Grid.Row="0" Background="Black" BorderBrush="White" BorderThickness="2" Foreground="White"/>

<TextBox Text="{Binding surname}" VerticalAlignment="Center" FontSize="20" MaxLength="20" Grid.Column="1" Grid.Row="1" Background="Black" BorderBrush="White" BorderThickness="2" Foreground="White"/>

<TextBox Text="{Binding patronymic}" VerticalAlignment="Center" FontSize="20" MaxLength="15" Grid.Column="1" Grid.Row="2" Background="Black" BorderBrush="White" BorderThickness="2" Foreground="White"/>

<TextBox Text="{Binding book\_name}" VerticalAlignment="Center" PreviewTextInput="TextBox\_PreviewTextInput" FontSize="20" Grid.Column="1" Grid.Row="3" Background="Black" BorderBrush="White" BorderThickness="2" Foreground="White"/>

<TextBox Text="{Binding genre}" VerticalAlignment="Center" PreviewTextInput="TextBox\_PreviewTextInput" FontSize="20" Grid.Column="1" Grid.Row="4" Background="Black" BorderBrush="White" BorderThickness="2" Foreground="White"/>

<TextBox Text="{Binding Pages}" PreviewTextInput="TextBox\_PreviewTextInput" VerticalAlignment="Center" FontSize="20" Grid.Column="1" Grid.Row="5" Background="Black" BorderBrush="White" BorderThickness="2" Foreground="White"/>

<TextBox Text="{Binding Cost}" VerticalAlignment="Center" FontSize="20" Grid.Column="1" Grid.Row="6" Background="Black" BorderBrush="White" BorderThickness="2" Foreground="White"/>

<TextBox Text="{Binding date\_issue}" VerticalAlignment="Center" FontSize="20" Grid.Column="1" Grid.Row="7" Background="Black" BorderBrush="White" BorderThickness="2" Foreground="White"/>

<TextBox Text="{Binding date\_admission}" VerticalAlignment="Center" FontSize="20" Grid.Column="1" Grid.Row="8" Background="Black" BorderBrush="White" BorderThickness="2" Foreground="White"/>

<TextBox Text="{Binding book\_availability}" VerticalAlignment="Center" FontSize="20" Grid.Column="1" Grid.Row="9" Background="Black" BorderBrush="White" BorderThickness="2" Foreground="White"/>

<Button Name="BtnSaveAndExit" Click="BtnSaveAndExit\_Click" Grid.ColumnSpan="2" Grid.Row="10" Width="300" Content="Сохранить" Background="Black" BorderBrush="White" BorderThickness="2" Foreground="White" />

</Grid>

</Page>

3.2 Программирование информационной системы

Код cs окна авторизации:

public MainWindow()

{

InitializeComponent();

Pass.IsEnabled = false;

Code.IsEnabled = false;

Log.Focus();

Refrsh.IsEnabled = false;

Enter.IsEnabled = false;

}

Код, который отвечает за проверку пользователя при входе в ИС, где идет подключение к базе данных :

private void Log\_KeyUp(object sender, KeyEventArgs e) // при освобождении клавиши

{

if (e.Key == Key.Enter)

{

using (var db = new BiblitekaEntities())

{

var number = db.Users.AsNoTracking().FirstOrDefault(m => m.login == // проверка логина в базе данных

Log.Text.Trim());

if (number == null)

{

MessageBox.Show("Неверный логин");

}

else

{

Pass.IsEnabled = true;

Log.IsEnabled = false;

Pass.Focus();

}

}

}

}

private void Pass\_KeyUp(object sender, KeyEventArgs e) // при освобожденнии клавиши

{

if (e.Key == Key.Enter)

{

using (var db = new BiblitekaEntities())

{

var password = db.Users.AsNoTracking().FirstOrDefault(m => m.login == // проверка пароля в базе данных

Log.Text.Trim() & (m.password == Pass.Password));

if (password == null)

{

MessageBox.Show("Неверный пароль");

}

else

{

Pass.IsEnabled = false;

gencode();

Code.Focus();

}

}

}

}

DispatcherTimer timer = new DispatcherTimer(); // таймер

string code;

private void gencode()

{

code = null;

Random random = new Random();

string[] massiveCharacters = new string[] { "1", "2", "3", "4", "5", "6", "7", "8" };

for (int i = 0; i < 4; i++)

code += massiveCharacters[random.Next(0, massiveCharacters.Length)];

if (MessageBox.Show(code.ToString(), "Code", MessageBoxButton.OK,

MessageBoxImage.Warning) == MessageBoxResult.OK)

{

timer.Interval = TimeSpan.FromSeconds(10);

timer.Tick += Timer\_Tick;

timer.Start();

Code.IsEnabled = true;

Enter.IsEnabled = true;

Refrsh.IsEnabled = true;

}

}

void Timer\_Tick(object sender, EventArgs e)

{

code = null;

Код, который фиксирует ограниченное время (10 сек.), выводит сообщение и сбрасывает рандомный код:

MessageBox.Show("Код сброшен. Повторите попытку");

timer.Stop();

}

private void Refrsh\_Click(object sender, RoutedEventArgs e)

{

timer.Stop();

gencode();

Code.Focus();

}

private void Enter\_Click(object sender, RoutedEventArgs e)

{

if (code == Code.Text)

{

timer.Stop();

using (var db = new BiblitekaEntities())

{

var user = db.Users.AsNoTracking().FirstOrDefault(u => u.login == Log.Text && u.password == Pass.Password);

if (user.Role == true)

{

AdminWindow kurs = new AdminWindow();

kurs.Show();

Application.Current.MainWindow.Close();

}

else

{

UserWindow kurs2 = new UserWindow();

kurs2.Show();

Application.Current.MainWindow.Close();

}

}

timer.Stop();

Код, в котором подтверждается системой авторизация пользователя:

MainWindow winavto = new MainWindow();

MessageBox.Show("Вы успешно вошли!"); // цикл срабатывает, если пользователь войдет под правильным логином и паролем, и затем успешно войдет в ИС

winavto.Show();

Close();

}

else

{

timer.Stop();

MessageBox.Show("Неверный код");

}

}

Код, отвечающий за отмену:

private void ClearBut\_Click(object sender, RoutedEventArgs e)

{

Log.Clear();

Pass.Clear();

Code.Clear();

Log.IsEnabled = true;

Pass.IsEnabled = false;

Code.IsEnabled = false;

Enter.IsEnabled = false;

Refrsh.IsEnabled = false;

Log.Focus();

}

}

}

Код cs главного окна:

public AdminWindow()

{

InitializeComponent();

Manager.MainFrame = MainFrame;

}

Код, который отвечает за переход при нажатии кнопки на “Книги”:

private void Books\_Click(object sender, RoutedEventArgs e)

{

MainFrame.Navigate(new BooksForEdit());

}

Код, который отвечает за переход при нажатии кнопки на “Дата выдачи и сбора книг”:

private void AllData\_Click(object sender, RoutedEventArgs e)

{

MainFrame.Navigate(new DataPeoples());

}

Код, который отвечает за переход при нажатии кнопки на “Выйти из аккаунта ”:

private void Exit\_Click(object sender, RoutedEventArgs e)

{

MainWindow mw1 = new MainWindow();

mw1.Show();

this.Close();

}

}

}

Код cs **“**Окно с данными о книгах”:

private void UpdateClients()

{

var currentBook = BiblitekaEntities.GetContext().Books.ToList();

currentBook = currentBook.Where(p => p.name.ToLower().Contains(TBoxSearch1.Text.ToLower())).ToList();

DGridBooks.ItemsSource = currentBook.OrderBy(p => p.name).ToList();

}

private void BtnEdit\_Click(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new Page1((sender as Button).DataContext as Book));

}

private void TBoxSearch1TextChanged(object sender, TextChangedEventArgs e)

{

UpdateClients();

}

private void BtnAdd\_Click(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new Page1(null));

}

private void BtnDel\_Click(object sender, RoutedEventArgs e)

{

var BookForRemoving = DGridBooks.SelectedItems.Cast<Book>().ToList();

if (MessageBox.Show($"Вы точно хотите удалить {BookForRemoving.Count()} книг?", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

try

{

BiblitekaEntities.GetContext().Books.RemoveRange(BookForRemoving);

BiblitekaEntities.GetContext().SaveChanges();

UpdateClients();

MessageBox.Show("Операция прошла успешно!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, ToString());

DGridBooks.ItemsSource = BiblitekaEntities.GetContext().Books.ToList();

}

}

}

private void DGridBooks\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

Код, который осуществляет экспорт отчета в ворд:

private void Otchet\_Click(object sender, RoutedEventArgs e)

{

var allRequest = BiblitekaEntities.GetContext().Books.ToList();

var application = new Word.Application();

Word.Document document = application.Documents.Add();

Word.Paragraph userParagraph = document.Paragraphs.Add();

Word.Range userRange = userParagraph.Range;

userRange.Text = "Книги";

userRange.InsertParagraphAfter();

Word.Paragraph tableParagraph = document.Paragraphs.Add();

Word.Range tableRange = tableParagraph.Range;

Word.Table paymentsTable = document.Tables.Add(tableRange, allRequest.Count() + 1, 8);

paymentsTable.Borders.InsideLineStyle = paymentsTable.Borders.OutsideLineStyle

= Word.WdLineStyle.wdLineStyleSingle;

= Word.WdCellVerticalAlignment.wdCellAlignVerticalCenter;

Word.Range cellRange;

Код, осуществляет обозревание объектов:

cellRange = paymentsTable.Cell(1, 1).Range;

cellRange.Text = "Название";

cellRange = paymentsTable.Cell(1, 2).Range;

cellRange.Text = "Автор";

cellRange = paymentsTable.Cell(1, 3).Range;

cellRange.Text = "Жанр";

cellRange = paymentsTable.Cell(1, 4).Range;

cellRange.Text = "Год";

cellRange = paymentsTable.Cell(1, 5).Range;

cellRange.Text = "Количество";

cellRange = paymentsTable.Cell(1, 6).Range;

cellRange.Text = "Наличие";

cellRange = paymentsTable.Cell(1, 7).Range;

cellRange.Text = "Стоимость";

cellRange = paymentsTable.Cell(1, 8).Range;

cellRange.Text = "Колл-во страниц";

paymentsTable.Rows[1].Range.Bold = 1;

paymentsTable.Rows[1].Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

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

{

var currentCategory = allRequest[i];

cellRange = paymentsTable.Cell(i + 2, 1).Range;

cellRange.Text = Convert.ToString(currentCategory.name);

cellRange.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

cellRange = paymentsTable.Cell(i + 2, 2).Range;

cellRange.Text = Convert.ToString(currentCategory.author);

cellRange = paymentsTable.Cell(i + 2, 3).Range;

cellRange.Text = Convert.ToString(currentCategory.genre);

cellRange = paymentsTable.Cell(i + 2, 4).Range;

cellRange.Text = Convert.ToString(currentCategory.year);

cellRange = paymentsTable.Cell(i + 2, 5).Range;

cellRange.Text = Convert.ToString(currentCategory.Quantity);

cellRange = paymentsTable.Cell(i + 2, 6).Range;

cellRange.Text = Convert.ToString(currentCategory.Cost);

cellRange = paymentsTable.Cell(i + 2, 8).Range;

cellRange.Text = Convert.ToString(currentCategory.Quantity);

}

application.Visible = true;

}

private void DGridBooks\_SelectionChanged\_1(object sender, SelectionChangedEventArgs e)

{

}

}

}

Код cs “Окно Добавление”:

private void BtnSaveAndExit\_Click(object sender, RoutedEventArgs e)

{

if (string.IsNullOrWhiteSpace(\_currentBook.name))

errors.AppendLine("Укажите название книги.");

if (string.IsNullOrWhiteSpace(\_currentBook.author))

errors.AppendLine("Укажите Автора книги.");

if (string.IsNullOrWhiteSpace(\_currentBook.genre))

errors.AppendLine("Укажите жанр книги.");

if (\_currentBook.year < 1)

errors.AppendLine("Укажите год написания книги.");

if (\_currentBook.Availability == null)

errors.AppendLine("Книги не существует? Тут надо ввести True(есть в наличии) False(нет в наличии).");

if (\_currentBook.Cost == null)

errors.AppendLine("Укажите стоимость книг.");

if (\_currentBook.Quantity == null)

errors.AppendLine("Укажите количество книг.");

if (\_currentBook.Pages == null)

errors.AppendLine("Укажите количество страниц книги.");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

// тут идет проверка на ввод данных

if (\_currentBook.id == 0)

BiblitekaEntities.GetContext().Books.Add(\_currentBook);

try

{

BiblitekaEntities.GetContext().SaveChanges();

MessageBox.Show("Информация сохранена!");

Manager.MainFrame.GoBack();

}

// окно с сообщением

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

private void TextBox\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

ЗАКЛЮЧЕНИЕ

В настоящее время нельзя представить ни одно предприятие, организацию, без ведения какой - либо системы. С каждым днем требуется все более спроектированные системы.

По итогам работы можно сказать, что было сделано хорошее проектирование информационной системы для библиотеки. Вычислительная техника все более широко используется не только для проектирования системы, но и в самых различных его сферах. Для реализации проекта были выполнены цели и задачи – разработка предпроектной части требований к информационной системе, спроектировать требования к информационной системе, разработать прототип информационной системы.

Автоматизированная информационная система «Библиотека» позволяет сформировать отчеты по читателям, книгам, авторам, задолжникам. Также в информационной системе имеются отчёты по списанию книги (различные причины).

Ссылка на репозиторий GitHub:

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Филип Джемпикс. Язык программирования C#/ Троесон Эндрю– Лондон: 2018.
2. Джозеф Албахари. C# 7.0. Справочник: полное описание языка / Бен Албахари– Лондон: 2016.

**Интернет-ресурсы**

1. Полное руководство по языку программирования С# 11 и платформе.// C# - .NET– Режим доступа: https://metanit.com/sharp/tutorial/
2. Справочник 'WPF+C#'// Stack Overflow. – Режим доступа: <https://itproger.com/course/csharp-wpf/4>
3. Учебник ‘C#// для начинающих’. – Режим доступа: <https://lesmatveev.narod.ru/knigi_proect/ci_sharp.pdf>
4. Документация ‘Enity Framework’ – Режим доступа: <https://learn.microsoft.com/ru-ru/ef/>
5. Справочник ‘ WpfHost’ – Режим доступа: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/app-development/hosting-wpf-applications?view=netframeworkdesktop-4.8