**КУРСОВАЯ РАБОТА**

**Тема:** **Разработка программного модуля системы**

**«Интернет магазин книг».**

**Специальность 09.02.07 Информационные системы и программирование**

|  |  |  |
| --- | --- | --- |
| **Выполнил студент(ка) группы** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Р.В. Капитонов** |
| **Руководитель** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **В.Ю. Назаров** |

**Москва 2023**

|  |
| --- |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  «РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И ГОСУДАРСТВЕННОЙ СЛУЖБЫ ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»  КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ |

**Задание на курсовой проект (работу)**

Дисциплина: МДК.01.01 Разработка программных модулей

Тема:

Специальность:09.02.07 Информационные системы и программирование

Группа: 31ИС-21

ФИО студента Капитонов Р.В.

ФИО руководителя Назаров В.Ю.

1. Проанализировать предметную область
2. Проанализировать готовые решения
3. Подготовить техническое задание
4. Подготовить план тестирования
5. Обосновать выбор инструментов и средств разработки
6. Описать реализацию технического задания
7. Выполнить тестирование
8. Подготовить инструкцию пользователю по работе с модулем

Задание выдано «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

Срок выполнения «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

Сроки защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Задание получил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc153195194)

[ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 8](#_Toc153195195)

[**1.1.** **Анализ предметной области** 8](#_Toc153195196)

[**1.2.** **Анализ существующих решений** 8](#_Toc153195197)

[**1.3.** **Постановка задач** 9](#_Toc153195198)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ 10](#_Toc153195199)

[**1.** **Описание среды разработки** 10](#_Toc153195200)

[**1.1.** **Выбор и описание программных инструментов** 10](#_Toc153195201)

[**1.2.** **Обоснование выбора инструментария по разработке** 11](#_Toc153195202)

[**2.** **Требования к функциям, выполняемым системой** 12](#_Toc153195203)

[**3.** **Приложение** 13](#_Toc153195204)

[**3.1.** **Модуль Classes** 13](#_Toc153195205)

[**3.2.** **Модуль BookDesktop** 14](#_Toc153195206)

[**3.3.** **Структура приложения** 15](#_Toc153195207)

[**3.4.** **Структура интерфейса приложения** 16](#_Toc153195208)

[ГЛАВА 3. РЕАЛИЗАЦИЯ ПРОЕКТА СИСТЕМЫ 18](#_Toc153195209)

[**1.1.** **Архитектура MVC** 18](#_Toc153195210)

[1.1.1. Взаимодействие между компонентами MVC 18](#_Toc153195211)

[1.1.2. Преимущества архитектуры MVC 18](#_Toc153195212)

[1.1.3. Недостатки архитектуры MVC 19](#_Toc153195213)

[**1.2.** **MVC в контексте .NET MAUI Blazor** 19](#_Toc153195214)

[1.3. Описание кодом функциональных узлов модуля BookDesktop 20](#_Toc153195215)

[1.3.1. Старт программы и подготовка 20](#_Toc153195216)

[1.3.2. Окно авторизации 20](#_Toc153195217)

[1.3.3. Каталог 21](#_Toc153195218)

[**1.4.** **Описание кодом функциональных узлов модуля Classes** 23](#_Toc153195219)

[1.4.1. AppDbContext 23](#_Toc153195220)

[1.4.2. Book 24](#_Toc153195221)

[1.4.3. Order 24](#_Toc153195222)

[1.4.4. User 25](#_Toc153195223)

[ГЛАВА 4. ТЕСТИРОВАНИЕ 27](#_Toc153195224)

[1. Демонстрация и результаты работы программного модуля 27](#_Toc153195225)

[1.1. Кейс «Неправильные данные авторизации» 27](#_Toc153195226)

[ЗАКЛЮЧЕНИЕ 29](#_Toc153195227)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 31](#_Toc153195228)

[ПРИЛОЖЕНИЯ 33](#_Toc153195229)

# ВВЕДЕНИЕ

Актуальность темы заключается в том, что интернет-магазины книг занимают свою нишу среди любителей литературы. Это связано с тем, что они предлагают широкий ассортимент книг, низкие цены и удобство покупки. Однако, для обеспечения эффективной работы интернет-магазина книг необходимо наличие развитой информационной системы.

В такой системе должны выполняться такие функции как:

* регистрация и авторизация пользователей;
* поиск книг;
* добавление книг в корзину;
* оформление заказа;
* оплата заказа;
* доставка заказа.

Разработка программного модуля системы «Интернет магазин книг» является актуальной задачей, поскольку позволит улучшить работу интернет-магазина и повысить его конкурентоспособность.

Разработанный программный модуль системы «Интернет магазин книг» может быть использован в различных интернет-магазинах книг. Он позволит автоматизировать основные бизнес-процессы, связанные с работой интернет-магазина, и повысить эффективность его работы.

Целью курсовой работы является разработка программного модуля системы «Интернет магазин книг».

Для достижения поставленной цели необходимо решить следующие задачи:

* провести анализ предметной области;
* провести анализ готовых решений;
* разработать логическую модель данных и проект модуля;
* реализовать программный модуль системы;
* провести тестирование программного модуля.

Объектом работы является процесс продажи книг.

Предметом работы является продажа книг через интернет-магазин.

В рамках курсовой работы будут рассмотрены:

* Авторизация пользователей;
* Обработка заказов;
* Управление книгами в системе;
* Просмотр действий других пользователей в системе.

# ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

* 1. **Особенности интернет-магазина**

дописать

Современные интернет-магазины, включая те, которые специализируются на продаже книг, являются важной частью розничной торговли. Они обладают особенностями, которые необходимо учесть при разработке программного модуля.

Ключевые аспекты включают:

* Ассортимент и каталог: Обширный каталог с четкими категориями для удобства выбора и навигации.
* Удобство покупки и навигация: Интуитивные интерфейсы, продвинутые системы фильтрации и рекомендации.
* Электронные и физические книги: Интеграция методов доставки и обработки заказов для обеих форматов.
* Технологические требования: Безопасность онлайн-платежей, мобильная совместимость и эффективная аналитика.

Этот обзор обеспечивает основополагающий взгляд на технические и функциональные аспекты, которые будут учтены при разработке программного модуля для системы "Интернет магазин книг".

* 1. **Анализ существующих решений**

Существует множество интернет-магазинов книг, каждый из которых предлагает уникальные решения и подходы. Некоторые из заметных проектов в этой области включают (см. приложение 1):

* Amazon Books
* Book Depository
* AbeBooks
* Google Play Books
* ЛитРес

Выводы по анализу

* 1. **Постановка задач**

Программный модуль системы "Интернет магазин книг" должен включать:

1. Функции управления каталогом
2. Функции обработки заказов
3. Проектирование:
   * Разработать дизайн пользовательского интерфейса для сотрудников магазина.
   * Выбрать архитектурные принципы и технологии для реализации бэкенда.
4. Разработка базы данных и бэкенда:
   * Создать структуру базы данных для хранения информации о книгах и заказах.
   * Реализовать бэкенд-логику, обеспечивающую взаимодействие с базой данных.
5. Функциональность управления товарами и заказами:
   * Разработать механизм управления описанием книг и их статусами.
   * Предоставить средства для отслеживания и управления заказами.
   * Интеграция Системы Оплаты:
   * Внедрить систему онлайн-оплаты для завершения заказов.
6. Административный интерфейс:
   * Создать административный интерфейс для управления заказами и товарами.
7. Тестирование и отладка:
   * Провести тестирование для выявления и исправления ошибок.

# ГЛАВА 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

1. **Описание среды разработки**
   1. **Выбор и описание программных инструментов**

Для разработки программного модуля системы «Интернет магазин книг» были выбраны следующие программные инструменты:

1. Среды разработки: Visual Studio 2022 и JetBrains Rider 2023.2.3
   * Описание: интегрированные среды разработки (IDE) от Microsoft и JetBrains соответственно, предназначены для создания широкого спектра приложений.
2. Язык программирования: C#
   * Описание: высокоуровневый язык программирования, разработанный Microsoft. Он обеспечивает возможность создания приложений для платформы .NET, используя объектно-ориентированный подход. C# предлагает разработчикам удобный и мощный инструментарий для создания разнообразных приложений, обеспечивая высокую производительность и эффективность в процессе разработки.
3. Платформа: .NET Framework 4.8.1
   * Описание: .NET Framework 4.8.1 представляет собой программную платформу, разработанную Microsoft, предназначенную для создания и выполнения приложений на языках программирования, таких как C#, VB.NET, и F#. Эта версия фреймворка обеспечивает среду для разработки широкого спектра приложений, включая веб-приложения, настольные приложения и службы.
4. Фреймворк EntityFramework:
   * Описание: представляет собой набор технологий в семействе технологий ADO.NET от Microsoft, предназначенных для разработки приложений, взаимодействующих с базами данных. Основная цель EF - упростить взаимодействие с базами данных, предоставляя разработчикам высокоуровневую объектно-ориентированную модель для работы с данными.
5. Фреймворк .NET MAUI с Blazor:
   * Описание: инструментарий для создания кроссплатформенных мобильных, настольных и веб-приложений с использованием языка программирования C# и платформы .NET. Сочетание .NET MAUI с Blazor позволяет использовать привычные технологии для создания веб-приложений и переносить их на мобильные и настольные платформы.
   1. **Обоснование выбора инструментария по разработке**

Причины выбора инструментария:

* Visual Studio 2022 – удобен в использовании и упрощает написание кода. Также имеет внушительный инструментарий и приятный интерфейс, который можно быстро понять.
* JetBrains Rider 2023.2.3 – мощный инструмент для написания кода. Очень удобен при навигации в коде и устранении технических проблем со сборкой проекта в целом.
* Язык программирования C# - множество библиотек и фреймворков, строгая типизация, удобное ООП, быстрая работа, много синтаксического сахара.
* .NET Framework 4.8.1 – последняя версия популярной платформы разработки. Обеспечивает инструментарий и скорость выполняемых приложений.
* EntityFramework – фреймворк, позволяющий работать с базами данных. Моделирует и осуществляет запросы к БД, предоставляя интерфейс в виде методов.
* .NET MAUI с Blazor – позволяет создавать кроссплатформенные приложения. Выбран из-за простого синтаксиса и удобной работой с пользовательским интерфейсом.

Эти элементы инструментария удовлетворяют требованиям для разработки приложения и дают возможность для дальнейшего расширения функционала.

1. **Требования к функциям, выполняемым системой**
2. Функции управления каталогом
   * Добавление книг: Возможность добавления новых книг в каталог, включая информацию о названии, авторе, жанре, цене и наличии.
   * Редактирование книг: Возможность изменения информации о книгах, уже находящихся в каталоге.
   * Фильтрация списка книг: пользователь должен иметь возможность фильтровать предлагаемые книги по жанрам.
3. Функции обработки заказов
   * Информация о заказах: хранение информации о заказах.
   * Управление заказами: изменение статуса заказов.
4. Функции корзины покупателя
   * Добавление в корзину: Возможность покупателями добавлять книги в корзину.
   * Редактирование корзины: Возможность изменения состава товаров в корзине.
   * Оформление заказа: Процедура оформления подписок.
5. Функции безопасности
   * Аутентификация и авторизация: Обеспечение безопасного доступа администраторов с использованием механизмов аутентификации и авторизации.

В приложении должны быть 2 строго разделённых модуля:

* Classes. Содержит классы, с которыми ведётся информационная работа.
* BookDesktop. Приложение с классами, которые его описывают.
  1. **Модуль Classes**

Рассмотрим диаграмму классов (см. рис.1), которые помогают вести информационную работу.

Изображение выглядит как текст, диаграмма, План, Параллельный

Автоматически созданное описание

Рисунок1 - Диаграмма классов

AppDbContext, класс реализующий работу с БД, хранится в этом же модуле для удобства работы с ним. Имеет ссылки на все остальные классы. Через EntityFramework создаёт базу данных с таблицами и связями, которые соответствуют таковым в коде модуля.

Некоторые неочевидные связи, такие как многие ко многим необходимо прописать явно (см. рис. 2).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2 - Описание связи многие ко многим

Почти все классы в модуле являются описанием таблиц, но для некоторых прописаны и переопределены методы для удобства работы с ними в приложении.

* 1. **Модуль BookDesktop**

Этот модуль написан на основе .NET MAUI фреймворка. При запуске открывается веб-приложение, которое является отдельным экземпляром браузера, который установлен в системе по умолчанию.

* Благодаря этому приложение само по себе является кроссплатформенным с проблемой исключительно в вёрстке.
* Также это позволит легко перейти на браузерную работу приложения при необходимости.

Связь с модулем Classes реализована через ссылку на сборку в модуле и директивы using там, где необходима прямая работа с классами первого модуля.

Класс AppDbContext здесь устанавливается как сервис. Для удобства проектирования приложения использовалась база данных SQLite, которая является файловой БД, т. е. хранится в одном файле вместе с описанием и самими данными, что удобно с небольшими базами или при отладке. Благодаря использованию EntityFramework, приложение можно быстро перенести на PostgreSQL.

Фреймворк .NET MAUI позволяет создавать веб-страницы с возможностью обработки действий пользователя. Синтаксис представляет собой HTML разметку с использованием «@», когда необходимо обратиться к переменным кода или использовать условные и цикличные структуры.

Также поддерживает «модели», что позволяет удобным образом создавать формы заполнения, устанавливая условия для длины поля и сообщения об ошибках, которые увидит пользователь при несоблюдении условий.

* 1. **Структура приложения**

Функционал, который предоставляет интерфейс ограничивается в зависимости от уровня прав пользователя (см. рис. 3).

Предусмотрено 3 уровня доступа:

1. Root - полный доступ. Предназначен для администратора системы.
2. Администратор. Предназначен для администратора занимающимся исключительно аналитикой. Не обладает правами на редактирование.
3. Сотрудник. Предназначен для сотрудника занимающимся каталогизацией книг и обработкой заказов.

Все действия логируются и доступны пользователям с доступом к аналитике.

Изображение выглядит как текст, диаграмма, зарисовка, рисунок

Автоматически созданное описаниеРисунок 3 - Диаграмма прецедентов

* 1. **Структура интерфейса приложения**

В первую очередь при запуске приложения пользователь попадает на страницу авторизации. После ввода и подтверждения программа запоминает авторизованного пользователя, как объект класса User из модуля Classes.

Пользователю с соответствующими правами будут доступны:

* Главная страница
* Каталог
* Логи
* Заказы
  + 1. Главная страница

На этой странице можно посмотреть основную информацию о пользователе, от имени которого производятся действия в программе. На этой же странице кнопка выхода, если есть необходимость ей воспользоваться.

* + 1. Каталог

Одна из основных страниц. Отображает все книги, которые есть в базе данных и основную информацию о них. Позволяет изменить определённую книгу, добавить новую или отфильтровать по жанрам.

Функции редактирования и добавления не доступны для администратора(аналитика).

* + 1. Логи

Страница аналитики. Отображает все действия, которые записаны системой. Записываются только изменения, которые повлияли на данные в базе данных. Для просмотра всех действий пользователей необходима модификация программы.

* + 1. Заказы

Вторая по важности страница. Представляет все заказы, которые есть в БД в виде карт. Позволяет изменять статус заказа.

Схема бд

И примеры интерфейса добавить

# ГЛАВА 3. РЕАЛИЗАЦИЯ ПРОЕКТА МОДУЛЯ СИСТЕМЫ

* 1. **Архитектура MVC**

При реализации интерфейса использовалась архитектура MVC (Model-View-Controller). Она подразумевает разделение обработки и работы с кодом на модули: модель, представление и контроллер.

Модель отвечает за хранение данных и бизнес-логику приложения. Она не зависит от представления и контроллера и может использоваться в других приложениях.

Представление отвечает за отображение данных модели пользователю. Оно не зависит от модели и контроллера и может быть использовано для отображения данных из других источников.

Контроллер отвечает за обработку пользовательских запросов и взаимодействие между моделью и представлением. Он зависит от модели и представления и может быть использован для обработки пользовательских запросов из других приложений.

* + 1. Взаимодействие между компонентами MVC

Пользователь взаимодействует с приложением через представление. Представление получает запрос от пользователя и передает его контроллеру. Контроллер обрабатывает запрос и вызывает соответствующие методы модели для получения данных. Данные модели передаются представлению, которое отображает их пользователю.

* + 1. Преимущества архитектуры MVC

Разделение кода на три независимых компонента упрощает разработку, тестирование и поддержку приложения.

Независимость компонентов позволяет легко изменять или расширять приложение.

* + 1. Недостатки архитектуры MVC

При неправильном использовании архитектура MVC может привести к снижению производительности приложения.

* 1. **MVC в контексте .NET MAUI Blazor**

Взаимодействие между компонентами MVC в контексте .NET MAUI Blazor осуществляется через свойства, события и методы компонентов.

**Свойства** компонентов MVC могут использоваться для передачи данных между компонентами. Модель может иметь свойство, которое содержит данные, которые должны быть отображены в представлении. Представление может получить доступ к этому свойству и использовать его для отображения данных пользователю.

**События** компонентов MVC могут использоваться для оповещения других компонентов о событиях, которые происходят в приложении. Например, представление может иметь событие, которое возникает, когда пользователь нажимает кнопку. Контроллер может подписаться на это событие и выполнить действие в ответ на нажатие кнопки.

**Методы** компонентов MVC могут использоваться для выполнения действий, которые влияют на приложение. Например, контроллер может иметь метод, который вызывается для получения данных от сервера. Модель может иметь метод, который выполняет бизнес-логику приложения.

**Представление** представляет собой компонент Blazor, который отображает указанные элементы.

## Описание кодом функциональных узлов модуля BookDesktop

Рассмотрим основные моменты:

* + 1. Старт программы и подготовка

Программа начинает своё выполнение с класса MauiProgram, где приложения собирается. Производится добавление сервисов и их подготовка.

После завершения подготовительных работ инициализируется интерфейс и появляется окно программы со страницей авторизации.

* + 1. Окно авторизации

Представляет собой форму с двумя полями. Ниже код страницы:

@page "/"  
@inject AuthService authService  
@inject NavigationManager navigation  
@using BookDesktop.Models  
@using BookDesktop.Service  
  
<div class="w-50 m-auto">  
 <h3>Авторизация</h3>  
 <**EditForm Model**="@Person" **OnValidSubmit**="@Submit">  
 <**DataAnnotationsValidator** />  
 <**ValidationSummary** />  
 <div class="mb-3">  
 <label for="exampleInputEmail1" class="form-label">Логин</label>  
 <**InputText** class="form-control" @**bind-Value**="Person!.Login" id="exampleInputEmail1" />  
 </div>  
 <div class="mb-3">  
 <label for="exampleInputPassword1" class="form-label">Пароль</label>  
 <**InputText** type="password" class="form-control" @**bind-Value**="Person!.Password" id="exampleInputPassword1" />  
 </div>  
 <button type="submit" class="btn btn-primary">Войти</button>  
 </**EditForm**>  
</div>  
  
@code  
{  
 Person Person = default!;  
  
 private void Submit()  
 {  
 if (Person.Login != "" && Person.Password != "")  
 {  
 if (!authService.Login(navigation, Person.Login!, Person.Password!))  
 {  
 Person.Login = String.Empty;  
 Person.Password = String.Empty;  
 InvokeAsync(StateHasChanged);  
 }  
 }  
 }  
  
 protected override void OnInitialized()  
 {  
  
 base.OnInitialized();  
 Person = new();  
 }  
  
}

* + 1. Каталог

Страница с перечислением всех книг в базе данных. Рядом на строке с каждой книгой есть кнопка для изменения. Ниже часть кода, отвечающая за обработку действий пользователя:

@code {  
 bool IsLoading = true;  
 List<Book> books = new();  
 List<Genre> genres = new();  
 List<Genre> selectedGenres = new();  
  
 async Task<List<Book>> GetBooksWithFilter()  
 {  
 if (selectedGenres.Count == 0)  
 return books;  
  
 var filteredBooks = await dbcontext.Books  
 .Include(x => x.Author)  
 .Where(book => book.Genres.Any(bg => selectedGenres.Contains(bg)))  
 .ToListAsync();  
  
 return filteredBooks;  
 }  
  
 async Task Load()  
 {  
 books = await dbcontext.Books  
 .Include(x => x.Author)  
 .ToListAsync();  
 genres = await dbcontext.Genres.ToListAsync();  
  
 IsLoading = false;  
 await InvokeAsync(StateHasChanged);  
 }  
  
 protected override async Task OnInitializedAsync()  
 {  
 await base.OnInitializedAsync();  
 await InvokeAsync(Load);  
 }  
  
 private void ChangeBook(Book book)  
 {  
 bookChangeService.ChangeBook(book);  
 }  
  
 private void ToggleGenreFilter(Genre genre)  
 {  
 if (selectedGenres.Contains(genre))  
 {  
 selectedGenres.Remove(genre);  
 }  
 else  
 {  
 selectedGenres.Add(genre);  
 }  
 InvokeAsync(StateHasChanged);  
 }  
  
}

* + 1. Страница просмотра логов

На нём отображается список всех действий пользователей, которые сохранены в базе данных.

@page "/log\_view"  
@using BookDesktop.Models  
@using BookDesktop.Service  
@using Classes  
@using Microsoft.EntityFrameworkCore  
@inject AppDbContext dbcontext  
@inject NavigationManager navigation  
@inject AuthService authService  
  
<**Menu**/>  
@if (IsLoading)  
{  
 <div>Загрузка</div>  
}  
else  
{  
 <h3>Логи</h3>  
 <table class="table table-bordered border-primary" aria-valuemin="200">  
 <tr>  
 <td>Исполнитель</td>  
 <td>Действие</td>  
 <td>Время</td>  
 </tr>  
 @foreach (var log in Logs)  
 {  
 <tr>  
 <td>@log.User?.Login</td>  
 <td>@log.Operation</td>  
 <td>@log.Created</td>  
 </tr>  
 }  
 </table>  
}  
  
@code {  
 bool IsLoading = true;  
 public List<Log> Logs { get; set; } = new();  
   
 protected override async Task OnInitializedAsync()  
 {  
 await base.OnInitializedAsync();  
  
 Logs = dbcontext.Logs.Include(x => x.User).ToList();  
  
 IsLoading = false;  
 await InvokeAsync(StateHasChanged);  
 }  
   
}

* 1. **Описание кодом функциональных узлов модуля Classes**

Состоит из 7 классов (6 функциональных, 1 служебный).

Рассмотрим основные:

* + 1. AppDbContext

using Microsoft.EntityFrameworkCore;  
  
namespace Classes;  
  
public class AppDbContext : DbContext  
{  
 public AppDbContext(DbContextOptions options) : base(options)  
 {  
 }  
  
 protected AppDbContext()  
 {  
 }  
  
 public DbSet<User> Users { get; set; }  
 public DbSet<Book> Books { get; set; }  
 public DbSet<Author> Authors { get; set; }  
 public DbSet<Genre> Genres { get; set; }  
 public DbSet<Order> Orders { get; set; }  
 public DbSet<Log> Logs { get; set; }  
  
 protected override void OnModelCreating(ModelBuilder modelBuilder)  
 {  
 modelBuilder.Entity<Book>()  
 .HasMany(b => b.Genres)  
 .WithMany(bg => bg.Books);  
  
 modelBuilder.Entity<Order>()  
 .HasMany(o => o.Books)  
 .WithMany(b => b.Orders);  
  
 modelBuilder.Entity<User>()  
 .HasMany(x => x.Logs)  
 .WithOne(b => b.User);  
 }  
}

* + 1. Book

using System.ComponentModel;  
using System.ComponentModel.DataAnnotations;  
using System.ComponentModel.DataAnnotations.Schema;  
  
namespace Classes;  
  
public class Book : INotifyPropertyChanged  
{  
 [Key, DatabaseGenerated(DatabaseGeneratedOption.**Identity**)]  
 public int Id { get; set; }  
 public string Name { get; set; } = "BookName";  
 public double Price { get; set; }  
 public int? AuthorId { get; set; }  
 [ForeignKey(nameof(AuthorId))]  
 public Author? Author { get; set; }  
 public byte[]? Image { get; set; }  
 public int Amount { get; set; } = 0;  
  
 public Book(int id, string name, Author author, double price, int amount) : this() =>  
 (this.Id, this.Name, this.Author, this.Price, this.Amount) = (id, name, author, price, amount);  
  
 public virtual List<Genre> Genres { get; set; } = new();  
 public virtual List<Order> Orders { get; set; } = new();  
 public User? LastRedactor { get; set; } = null;  
  
 public Book()  
 {  
   
 }  
  
 public event PropertyChangedEventHandler? PropertyChanged;  
  
 protected virtual void Dirty(string propertyName)  
 {  
 PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Id)));  
 PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Name)));  
 PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Author)));  
 PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Price)));  
 PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Genres)));  
 }  
}

* + 1. Order

using System.ComponentModel.DataAnnotations;  
using System.ComponentModel.DataAnnotations.Schema;  
  
namespace Classes;  
  
public enum OrderStatus  
{  
 New,  
 **Approved**,  
 **InDeliver**,  
 **Completed**,  
 **Cancelled**}  
  
public class Order  
{  
 [Key, DatabaseGenerated(DatabaseGeneratedOption.**Identity**)]  
 public int Id { get; set; }  
 public virtual List<Book> Books { get; set; } = new();  
 public int PerformerId { get; set; }  
 [ForeignKey(nameof(PerformerId))]  
 public User? Performer { get; set; }  
 public int LastRedactedId { get; set; }  
 [ForeignKey(nameof(LastRedactedId))]  
 public User? LastRedacted { get; set; }  
 public OrderStatus orderStatus { get; set; }  
 public Order()  
 {  
   
 }  
  
 public Order(OrderStatus status,int UserId=1, params Book[] books) : this()  
 {  
 this.Books = books.ToList();  
 this.orderStatus = status;  
 this.LastRedactedId = UserId;  
 this.PerformerId = UserId;  
 }  
}

* + 1. User

using System.ComponentModel;  
using System.ComponentModel.DataAnnotations;  
using System.ComponentModel.DataAnnotations.Schema;   
  
namespace Classes;  
  
public class User : INotifyPropertyChanged  
{  
 [Key, DatabaseGenerated(DatabaseGeneratedOption.**Identity**)]  
 public int Id { get; set; }  
 public string? Login { get; set; }  
 public string? Email { get; set; }  
 public string? Password { get; set; }  
 public int PermissionLvl { get; set; }  
 public User(int id, string name, string email = "", string password = "") : this() =>   
 (this.Id, this.Login, this.Email, this.Password) = (id, name, email, password);  
  
 public virtual List<Log> Logs { get; set; } = new();  
  
 public User()  
 {  
   
 }  
  
 public override string ToString()  
 {  
 return PermissionLvl switch  
 {  
 0 => "root",  
 1 => "Администратор",  
 2 => "Сотрудник",  
 \_ => "???"  
 };  
 }  
  
 public event PropertyChangedEventHandler? PropertyChanged;  
  
 protected virtual void Dirty(string propertyName)  
 {  
 PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Id)));  
 PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Login)));  
 PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Email)));  
 PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Password)));  
 PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(PermissionLvl)));  
 }  
}

# ГЛАВА 4. ТЕСТИРОВАНИЕ

1. Демонстрация и результаты работы программного модуля

Тестировать такую программу трудно в виду отсутствия взаимодействия с файлами, потому ограничимся тестами на данные авторизации и отсутствие подключения к базе данных.

* 1. Кейс «Неправильные данные авторизации»

В рамках кейса необходимо ввести неверные данные авторизации и посмотреть как отреагирует приложение.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описаниеВ качестве входных данных возьмём «Test1» как логин и пароль.

Рис. 4. Неверные данные авторизации

Результат – программа очистила поля и не авторизовала пользователя.

* 1. Кейс «Отсутствие подключения к базе данных»

Чтобы протестировать подобное я удалю файл базы данных, т. к. приложение использует SQLite

Программа создала новый файл базы данных и заполнила его тестовыми данными.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рис. 5. Содержимое таблицы Books после удаления базы данных

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были достигнуты следующие результаты:

* Проведен анализ предметной области, в результате которого была разработана концептуальная модель предметной области.
* Разработана логическая модель данных, которая отражает структуру данных, хранящихся в системе.
* Разработан программный модуль системы «Интернет магазин книг», который реализует следующие функции:
  + Авторизация пользователей;
  + Добавление книг в систему;
  + Изменение книг в системе;
  + Изменение статуса заказов;
  + Просмотр действий других пользователей.
* Проведено тестирование программного модуля, в результате которого было выявлено и устранено ряд ошибок.

Разработанный программный модуль системы «Интернет магазин книг» является решением для автоматизации внутренних бизнес-процессов интернет-магазина книг. Он позволяет обеспечить удобную и эффективную работу интернет-магазина, повысить его конкурентоспособность и привлечь новых клиентов.

Для внедрения разработанного программного модуля на объекте необходимо выполнить следующие действия:

* Установить программное обеспечение на рабочие компьютеры интернет-магазина.
* Обучить сотрудников интернет-магазина работе с программным обеспечением.
* Провести работу по адаптации продукта к системе интернет-магазина если потребуется.

Реализация этих действий позволит интернет-магазину книг повысить эффективность своей работы и улучшить качество обслуживания клиентов.

**Выводы и предложения**

В ходе выполнения курсовой работы были сделаны следующие выводы:

* Разработка программного модуля системы «Интернет магазин книг» является актуальной и перспективной задачей.
* Разработанный программный модуль является эффективным решением для автоматизации внутренних бизнес-процессов интернет-магазина книг.

На основе полученных результатов работы предлагается следующие предложения:

* Разработать дополнительные функции для программного модуля, такие как:
  + Возможность регистрации новых пользователей;
  + Улучшение работы с заказами;
  + Добавить поиск по книгам в окне заказов и книг по названиям;
  + Возможность интеграции системы с другими системами интернет-магазина.
* Провести анализ эффективности работы разработанного программного модуля в реальных условиях.

Реализация этих предложений позволит улучшить функциональность программного модуля и повысить его эффективность.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Нормативно-правовые источники

ГОСТ

Учебники, учебные пособия, статьи

Интернет-источники

1. URL: https://learn.microsoft.com/en-us/ef/core/get-started/overview/first-app?tabs=netcore-cli (03.12.2023)
2. URL: https://getbootstrap.com/ (11.12.2023)
3. URL: https://avaloniaui.net/ (21.11.2023)

Приложение 1

Таблица 1 Анализ существующих решений

|  |  |  |  |
| --- | --- | --- | --- |
| Магазин | Особенности | Преимущества | Ограничения |
| Amazon Books | Огромный каталог, электронные книги (Kindle), разнообразные сервисы | Широкий выбор товаров, интеграция с Amazon | Проблемы с поддержкой малых издательств |
| Book Depository | Международная доставка без платы за доставку, обширный международный каталог | Доступность для клиентов по всему миру, бесплатная международная доставка | Отсутствие электронных книг |
| AbeBooks | Маркетплейс для редких и старых книг, независимые продавцы | Уникальные коллекции, возможность найти редкие издания | Ограниченный выбор новых изданий |
| Google Play Books | Электронные книги для устройств Android, интеграция с экосистемой Google | Удобство использования на устройствах Android, синхронизация с облачным хранилищем Google | Ограниченное влияние вне экосистемы Android |
| ЛитРес | Российская платформа для электронных и аудиокниг, уникальные решения для российского рынка | Широкий выбор российских и зарубежных произведений, приспособленность к особенностям российского рынка | Возможные ограничения для международных клиентов |

Приложение 2

План тестирования