

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

**«ИНФОРМАЦИОННАЯ СИСТЕМА
УПРАВЛЕНИЯ ПЕРСОНАЛЬНЫМИ
ПОДБОРКАМИ ЛИТЕРАТУРНЫХ
ПРОИЗВЕДЕНИЙ»**

Студент С Советкин Даниил Дмитриевич
(подпись, дата) 25.05.2026 (ФИО полностью)

Руководитель А.С. Бычков А.С.
(подпись, дата) 25.05.2026 (фамилия, инициалы)

Нормоконтролёр А.С. Бычков А.С.
(подпись, дата) 25.05.2026 (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026г. № 14)

Заведующий кафедрой М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от
11 июня 2026г. № 5)

Секретарь ГЭК Е. Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«07» 04 2026 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Советкин Даниил Дмитриевич гр. 22BVB1 факультета ФВТ

2. Тема работы Информационная система управления персональными подборками литературных произведений

Тема утверждена приказом ИТУ № 347/о от "07" 04 2026 г.

3. Руководитель бакалаврской работы Бычков А.С.

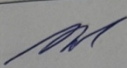
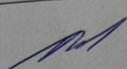
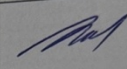
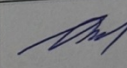
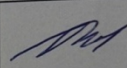
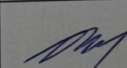
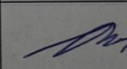
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать информационную систему управления персональными подборками литературных произведений для автоматизации процесса поиска, отбора и систематизации книжного контента. Для того чтобы достичь максимальной эффективности при работе с книжными коллекциями, а также снизить риск потери времени на поиск литературы, часто применяют системы персонализированных рекомендаций, которые позволяют автоматически подбирать книги на основе предпочтений пользователя. Разработанная программа для ускорения процесса поиска и отбора книжного контента позволяет уменьшить временные затраты пользователя, избавиться от необходимости ручного просмотра большого количества неподходящих изданий и сконцентрироваться на интересующей литературе, отслеживать всю необходимую информацию и сокращать время на выполнение простых операций по поиску и систематизации книжного фонда.

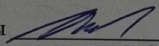
5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: изучить существующие информационные системы, выделить их достоинства и недостатки. Выявить потребности пользователей. Выбрать язык программирования, среду разработки, изучить вопрос электротехники и схемотехники. Определить архитектуру приложения и вспомогательные средства разработки. Изучить функционал Visual Studio для разработки информационную систему управления персональными подборками литературных произведений.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	32	08.04.2026 – 15.04.2026	
2	Анализ требований	32	16.04.2026 – 23.04.2026	
3	Создание макета графического интерфейса	30	24.04.2026 – 01.05.2026	
4	Выбор технологий, применяемых при проектировании и разработке	40	02.05.2026 – 12.05.2026	
5	Реализация графического интерфейса	66	13.05.2026 – 20.05.2026	
6	Разработка основной части программы	70	21.05.2026 – 25.05.2026	
7	Оформление пояснительной записки	54	26.05.2026 – 30.05.2026	

Дата выдачи задания "07" апреля 2026 г.

Руководитель бакалаврской работы  Бычков А.С.
(подпись, дата)

Задание к исполнению принял студент  Советкин Д.Д.
(подпись, дата)

Работу к защите допустить
Декан факультета  09.06.2026 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 156 листов, 42 рисунка, 2 таблицы, 7 фрагментов листинга, 7 источников, 3 приложения.

ПЕРСОНАЛИЗАЦИЯ, WEB, API, ПРОГРАММА, БАЗА ДАННЫХ, УПРАВЛЕНИЕ, ПОЛЬЗОВАТЕЛЬ, РЕКОМЕНДАЦИЯ, ПРОИЗВЕДЕНИЕ, ПОДБОРКА, ФИЛЬТРАЦИЯ

Цель работы – разработать информационную систему управления персональными подборками литературных произведений.

Одним из ключевых решений для реализации дипломного проекта является информационная система на основе настольного приложения Windows Forms, обеспечивающая выполнение следующих функций: просмотр и редактирование личных данных, управление коллекцией книг, предоставление персонализированных рекомендаций в зависимости от предпочтений пользователя на основе коллаборативной фильтрации, а также экспорт списка избранных книг в PDF – документ.

Данная программа полностью соответствует цели данной выпускной квалификационной работы знания в области работы с реляционными базами данных SQLite, опыт интеграции с внешними API, а также умения в создании графических пользовательских интерфейсов на платформе Windows Forms. Данная программа протестирована и уже эксплуатируется.

Изм.	Лист	№ докум.	Подпись	Дата	Лит.			Лист	Листов
Разраб.		Саветкин Д.Д.		25.05.20	Информационная система управления персональными подборками литературных произведений Пояснительная записка			2	158
Провер.		Бычков А.С.		25.05.20					
Н. Контр		Бычков А.С.		25.05.20					
Утверд.		Митрахан М.А.		25.05.20	ФВТ, каф. ВТ			гр. 22BBB1	

ПГУ1.090301.11.001 ПЗ

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	8
1 Информационная система управления персональными подборками литературных произведений.....	10
1.1 Анализ предметной области.....	10
1.2 Обзор аналогов.....	11
1.3 Анализ требований к разработке.....	14
1.3.1 Требования к операционной системе.....	17
1.3.2 Требования к процессору.....	18
1.3.3 Требования к оперативной памяти.....	18
1.3.4 Требования к дисковому пространству.....	18
1.3.5 Требования к разрешению экрана.....	19
1.3.6 Требования к сетевому подключению.....	19
1.4 Выбор архитектуры системы.....	20
1.5 Выбор языка программирования.....	22
1.6 Выбор среды разработки.....	23
1.7 Разработка управляющей программы.....	32
1.8 Графический интерфейс программы.....	36
1.8.1 Окно аутентификации пользователя.....	36
1.8.2 Окно регистрации нового пользователя.....	38
1.8.3 Окно восстановления пароля.....	39
1.8.4 Вкладка поиска в главном окне приложения.....	40
1.8.5 Вкладка рекомендаций.....	42
1.8.6 Вкладка избранного.....	44

1.8.7 Вкладка профиля и статистики	45
1.9 Тестирование и анализ результатов	48
1.9.1 План тестирования	48
1.9.2 Результаты тестирования	50
1.9.3 Вывод по результатам тестирования	59
2 Требования безопасности по охране труда при работе с ПЭВМ	60
2.1 Общие требования безопасности	60
2.2 Электробезопасность	62
2.3 Пожарная безопасность	64
2.4 Режим труда и отдыха	68
Заключение	70
Список использованных источников	71
Приложение А— UML диаграммы и схемы	73
Приложение Б— Листинг программы	81
Приложение Б— Листинг программы	81
Приложение В — Презентация	158

Перечень сокращений и обозначений

В данной работе используются такие сокращения, как:

– БД (база данных) — структурированная совокупность данных, хранящихся в соответствии с определённой схемой. В работе используется для хранения информации о пользователях, избранных книгах и кэшированных обложках;

– ГБ (гигабайт) — единица измерения количества информации, равная 1024 мегабайтам (1 073 741 824 байт). Используется для определения требований к дисковому пространству;

– КБ (килобайт) — единица измерения количества информации, равная 1024 байтам. Используется для оценки объёма кэшированных изображений и размера записей в базе данных;

– МБ (мегабайт) — единица измерения количества информации, равная 1024 килобайтам (1 048 576 байт). Используется для определения требований к оперативной памяти и объёму установочного пакета;

– ОС (операционная система) — комплекс программных средств, обеспечивающий управление аппаратными ресурсами компьютера и предоставляющий интерфейс для взаимодействия с пользователем. Целевой платформой является семейство Microsoft Windows;

– ПО (программное обеспечение) — совокупность программ, предназначенных для выполнения определённых задач на вычислительной технике. Разработанное приложение является примером прикладного ПО;

– ПЭВМ (персональная электронно – вычислительная машина) — устройство, предназначенное для автоматической обработки информации с использованием электронных компонентов. Является целевой платформой для разрабатываемого приложения;

– СУБД (система управления базами данных) — комплекс программных средств, предназначенный для создания, управления и обработки баз данных. В работе в качестве СУБД используется SQLite;

					ПГУ1.090301.11.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

– API (Application Programming Interface) — упрощает процесс интеграции приложения с внешними сервисами, абстрагируя базовую реализацию и предоставляя только необходимые методы для взаимодействия. В работе используется для получения данных о книгах из открытого каталога OpenLibrary;

– BLOB (Binary Large Object) — тип данных, предназначенный для хранения двоичной информации, такой как изображения, аудио– или видеофайлы. В работе используется для сохранения аватаров пользователей и кэширования обложек книг в базе данных;

– HTML (HyperText Markup Language) — стандартизированный язык разметки документов для просмотра веб – страниц в браузере. Используется API OpenLibrary для предоставления веб – интерфейса;

– HTTP (HyperText Transfer Protocol) — протокол прикладного уровня, используемый для передачи данных между клиентом и веб – сервером. Применяется для отправки запросов к OpenLibrary API;

– IDE (Integrated Development Environment) — программный комплекс, предоставляющий инструменты для написания, отладки и компиляции программного кода. В работе используется Microsoft Visual Studio 2022;

– JSON (JavaScript Object Notation) — текстовый формат обмена данными, основанный на синтаксисе JavaScript, обеспечивающий удобное структурированное представление информации. Используется для парсинга ответов API OpenLibrary;

– PDF (Portable Document Format) — кроссплатформенный формат электронных документов, разработанный компанией Adobe, обеспечивающий корректное отображение содержимого независимо от устройства и операционной системы. Используется в работе для экспорта списка избранных книг;

– UML (Unified Modeling Language) — язык графического моделирования, предназначенный для описания, визуализации, проектирования и документирования программных систем. Используется для построения диаграмм вариантов использования, последовательностей и развёртывания.

					ПГУ1.090301.11.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Необходимость быстрой и точной идентификации автора литературного произведения имеет место практически в любой сфере деятельности человека, связанной с работой с текстами. В зависимости от профессиональных задач — будь то написание научной работы, библиографическое описание фондов или просто поиск интересующей книги для чтения — требуется оперативно получать достоверные сведения об авторстве. В современном мире, характеризующемся стремительным ростом объема электронного контента, ручной поиск этой информации в разнородных источниках становится трудоемкой рутинной задачей, которую целесообразно автоматизировать с применением технологических решений.

Тема цифровых библиотек и автоматизации библиографического поиска приобретает все большую актуальность в эпоху информационного изобилия. В связи с этим возникает ряд проблем, связанных с достоверностью метаданных, их фрагментацией по различным источникам и сложностью оперативного получения структурированной информации об авторе. Ведь именно точная идентификация авторства позволяет корректно классифицировать произведение, оценить его релевантность и избежать ошибок при цитировании.

Чтобы избежать временных потерь и ошибок, обусловленных человеческим фактором, эффективным решением является создание программного приложения, которое позволит автоматизировать процесс поиска автора по названию книги, структурировать полученные данные и предоставить пользователю возможность сохранять результаты для последующего использования.

Эффективность такого инструмента напрямую зависит от оперативности доступа к авторитетным источникам данных и качества их обработки. В условиях, когда информация о книгах рассредоточена по множеству веб –

платформ и библиотечных каталогов, возникает задача ее агрегации и унифицированного представления. Существующая практика ручного копирования этих данных из браузера в текстовые документы или таблицы является узким местом, снижающим общую эффективность работы с библиографической информацией.

Целью данной работы является разработка инструмента, автоматизирующего процесс поиска и первичной обработки библиографических метаданных. Создаваемая программа реализует парсинг данных с открытого библиотечного API по названию книги, их структурированное отображение пользователю и автоматическое сохранение выбранных записей в персонализированный список избранного. Такой подход позволяет унифицировать процесс получения информации об авторе, исключить влияние человеческого фактора на этапе сбора данных и предоставить пользователю консолидированную информацию в привычном и удобном для дальнейшей обработки формате.

1 Информационная система управления персональными подборками литературных произведений

1.1 Анализ предметной области

Предметная область данной работы охватывает процессы информационного поиска и автоматической идентификации библиографических метаданных, ассоциированных с литературным произведением. Основным источником данных выступают открытые электронные библиотечные каталоги, предоставляющие программный интерфейс API для доступа к своим базам данных. Пользовательское взаимодействие в данной области, как правило, осуществляется через веб-интерфейсы, что не всегда удобно при необходимости быстрого, пакетного или автоматизированного получения информации.

Ключевым аспектом функционирования систем подобного класса является способность корректно интерпретировать ответы от внешних сервисов. В качестве основного поставщика библиографических данных в работе используется открытое ПО, которое позволяет выполнять поиск по широкому спектру параметров, включая название произведения, имя автора или ключевые слова, и получать структурированный JSON – ответ, содержащий такие поля, как автор, год первой публикации, количество страниц, идентификатор обложки и аннотация.

Анализ пользовательских сценариев показывает, что целевая аудитория программного продукта заинтересована не только в получении базовой информации об издании, но и в удобном представлении сопутствующих данных, возможности их сохранения, категоризации и последующего экспорта. Таким образом, эффективное решение должно сочетать в себе функции клиента для получения данных из внешнего источника, средства структурированного отображения информации и локального менеджера,

обеспечивающего хранение персонального списка избранного и его аналитическую обработку.

Актуальность разработки подобного инструмента приобретает в контексте образовательных учреждений и небольших библиотек, функционирующих в условиях ограниченного бюджета. В этих условиях разработка специализированного приложения, ориентированного именно на поиск, систематизацию и персональный учёт библиографической информации с использованием бесплатных открытых API и легковесных локальных баз данных, представляется наиболее рациональным и оправданным решением.

1.2 Обзор аналогов

В ходе анализа предметной области были рассмотрены существующие программные решения и веб – сервисы, решающие схожие задачи идентификации книг и их авторов. Сравнение проводилось по следующим критериям: способ получения данных, функциональность, возможность персонализации и наличие открытого программного интерфейса для интеграции со сторонними приложениями.

ЛитРес – один из самых крупных российских сервисов электронных книг, предоставляющий доступ к более чем одному миллиону произведений. Сервис предлагает развитую экосистему: мобильные приложения с возможностью чтения и прослушивания, синхронизацию между устройствами, персонализированные рекомендации и программу лояльности. Сайт проекта представлен на рисунке 1.

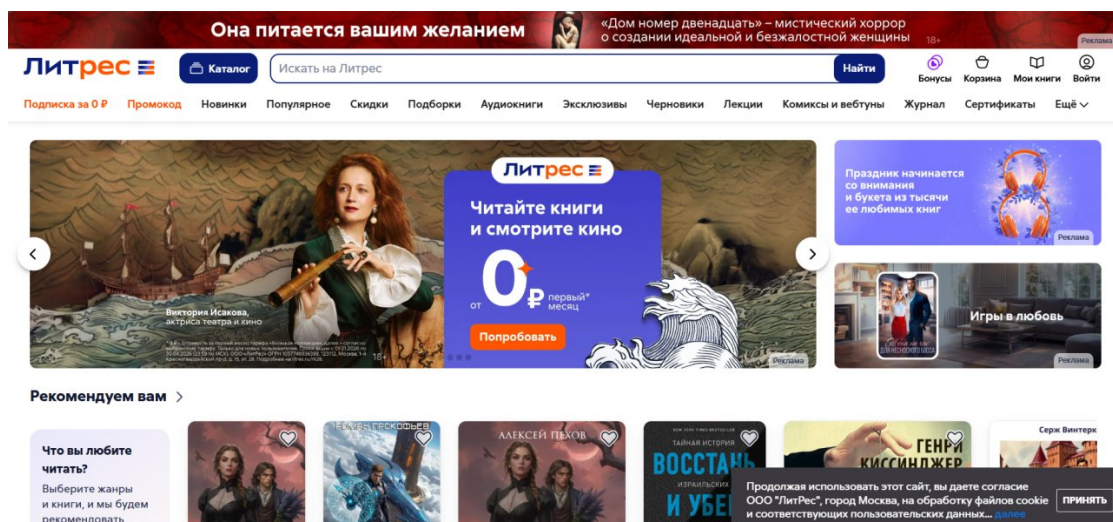


Рисунок 1 – Литрес

Альдебаран — онлайн – библиотека, предлагающая пользователям доступ к значительному собранию текстов различной тематики и жанров. Сайт имеет простую структуру каталога с постраничной навигацией и карточками книг, содержащими название, автора и краткое описание. Интерфейс показан на рисунке 2.

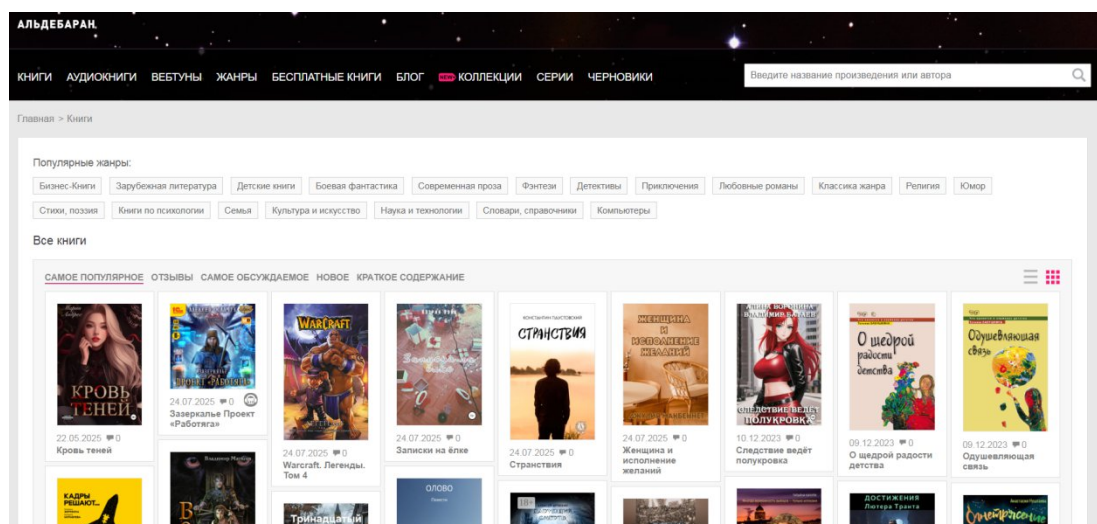


Рисунок 2 — Goodreads

Яндекс Книги — крупнейший книжный сервис в России, входящий в экосистему Яндекс Плюс. По состоянию на 2026 год каталог сервиса насчитывает более 250 тысяч электронных и аудиокниг. Сервис активно

внедряет технологии искусственного интеллекта: функция «Ранее в книге» на базе YandexGPT позволяет напоминать содержание предыдущих глав, а «Виртуальный рассказчик» озвучивает текстовые книги с естественными интонациями. Приложение доступно для iOS и Android, поддерживает создание персональных полок, сохранение цитат и заметок, а также синхронизацию между устройствами[1]. На рисунке 3 изображен сайт проекта.

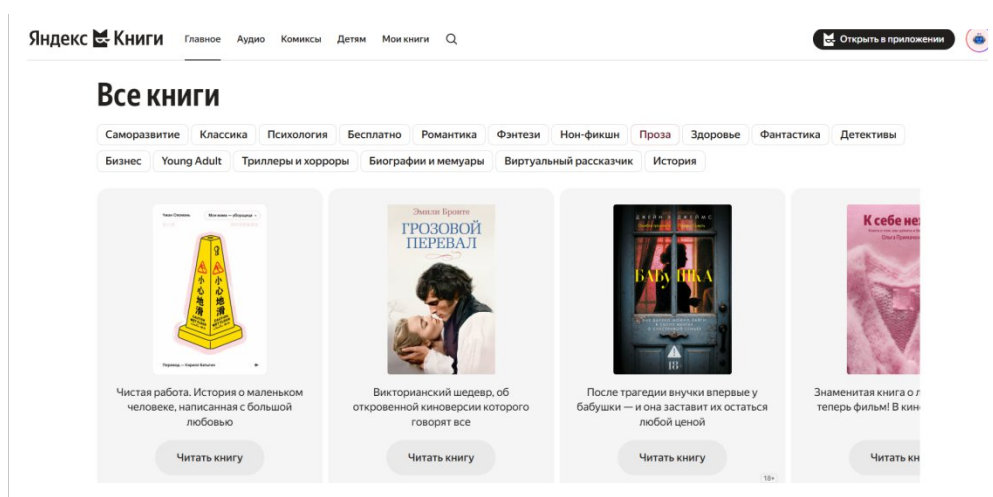


Рисунок 3 — Яндекс Книги

Сравнительный анализ преимуществ и недостатков рассмотренных решений представлен в таблице 1.

Таблица 1—Сравнительная таблица аналогов

Производитель	Преимущества	Недостатки
ЛитРес (https://www.litres.ru/)	Крупнейший русскоязычный каталог, наличие API для партнеров, развитая экосистема приложений, интеграция с библиотеками	API доступен преимущественно коммерческим партнерам; для автоматизированного использования непригоден

Производитель	Преимущества	Недостатки
Альдебаран (https://aldebaran.one/knigi/)	Открытый бесплатный доступ, широкий выбор произведений, не требует подписки для чтения	Отсутствие API; данные доступны только в виде HTML – страниц; качество метаданных не гарантировано
Яндекс Книги (https://books.yandex.ru/)	Интеграция с Яндекс Плюс (сотни тысяч книг), удобное приложение, ИИ –функции	Полностью закрытый сервис; нет API; доступ только по платной подписке

Все рассмотренные решения имеют существенные ограничения с точки зрения их использования в качестве источника данных для автоматизированного приложения. Коммерческие сервисы являются закрытыми платформами без публичного API. Ресурс с открытым доступом не предоставляет программного интерфейса, что требует применения технологий веб – парсинга. Это подтверждает актуальность разработки собственного инструмента, который мог бы интегрироваться с открытыми источниками данных и предоставлять пользователю унифицированный и удобный интерфейс для поиска информации об авторе книги, агрегируя данные из доступных источников.

1.3 Анализ требований к разработке

На основе анализа предметной области и выявленных недостатков существующих аналогов были сформулированы функциональные и нефункциональные требования к разрабатываемому приложению.

Функциональные требования:

– аутентификация и регистрация пользователя: Система должна обеспечивать разделение доступа на основе учетных записей для персонализации хранения данных. Требуется реализовать функции входа в систему и регистрации нового пользователя с указанием возраста для последующей фильтрации контент, а также с возможностью восстановления пароля;

– поиск библиографической информации: Приложение должно предоставлять графический интерфейс для ввода поискового запроса, отправлять асинхронный запрос к открытой библиотеке и получать список результатов в формате JSON;

– отображение результатов поиска: Найденные книги должны отображаться в структурированном табличном виде с указанием названия, автора, года первой публикации и среднего рейтинга читателей. При отсутствии обложки должна автоматически генерироваться графическая заглушка с соответствующей надписью;

– просмотр детальной информации: при выборе конкретной книги из списка результатов приложение должно отображать расширенную информацию о ней, включая полное название, автора, год издания, рейтинг, аннотацию и прямую ссылку на ресурс Open Library, а также асинхронно загружать и кэшировать обложку книги в локальной базе данных;

– управление избранным: Авторизованный пользователь должен иметь возможность добавлять выбранные книги в персональный список «Избранное» как из результатов поиска, так и из панели рекомендаций. Необходимо реализовать просмотр списка избранного, удаление отдельных записей или полную очистку списка. Данные должны сохраняться в локальной базе данных SQLite между сеансами работы;

– система рекомендаций: Приложение должно анализировать списки избранного всех пользователей и на основе алгоритма коллаборативной

фильтрации предлагать пользователю книги, добавленные читателями со схожими предпочтениями (при наличии не менее двух совпадающих книг);

- возрастная фильтрация контента: при регистрации пользователь указывает свой возраст. Для пользователей младше 18 лет система должна автоматически исключать из поисковой выдачи и рекомендаций книги, содержащие маркеры «взрослого» контента, на основе анализа тематических рубрик;

- визуализация статистики: В личном кабинете пользователя должно отображаться графическое представление распределения книг по дате добавления в избранное;

- экспорт библиографического списка: Пользователь должен иметь возможность экспортировать свой список избранного в файл формата PDF, оформленный в табличном виде, для дальнейшего использования при подготовке списков литературы;

- настройка интерфейса: Пользователь должен иметь возможность изменять цветовую тему оформления приложения (светлая, тёмная, лесная) с немедленным применением изменений и сохранением выбора в его профиле.

Исходя из целевого назначения проекта, были сформированы диаграммы прецедентов для неавторизованного и авторизованного пользователей, изображённые на рисунках 4 и 5.



Рисунок 4 – Диаграмма прецедентов неавторизованного пользователя

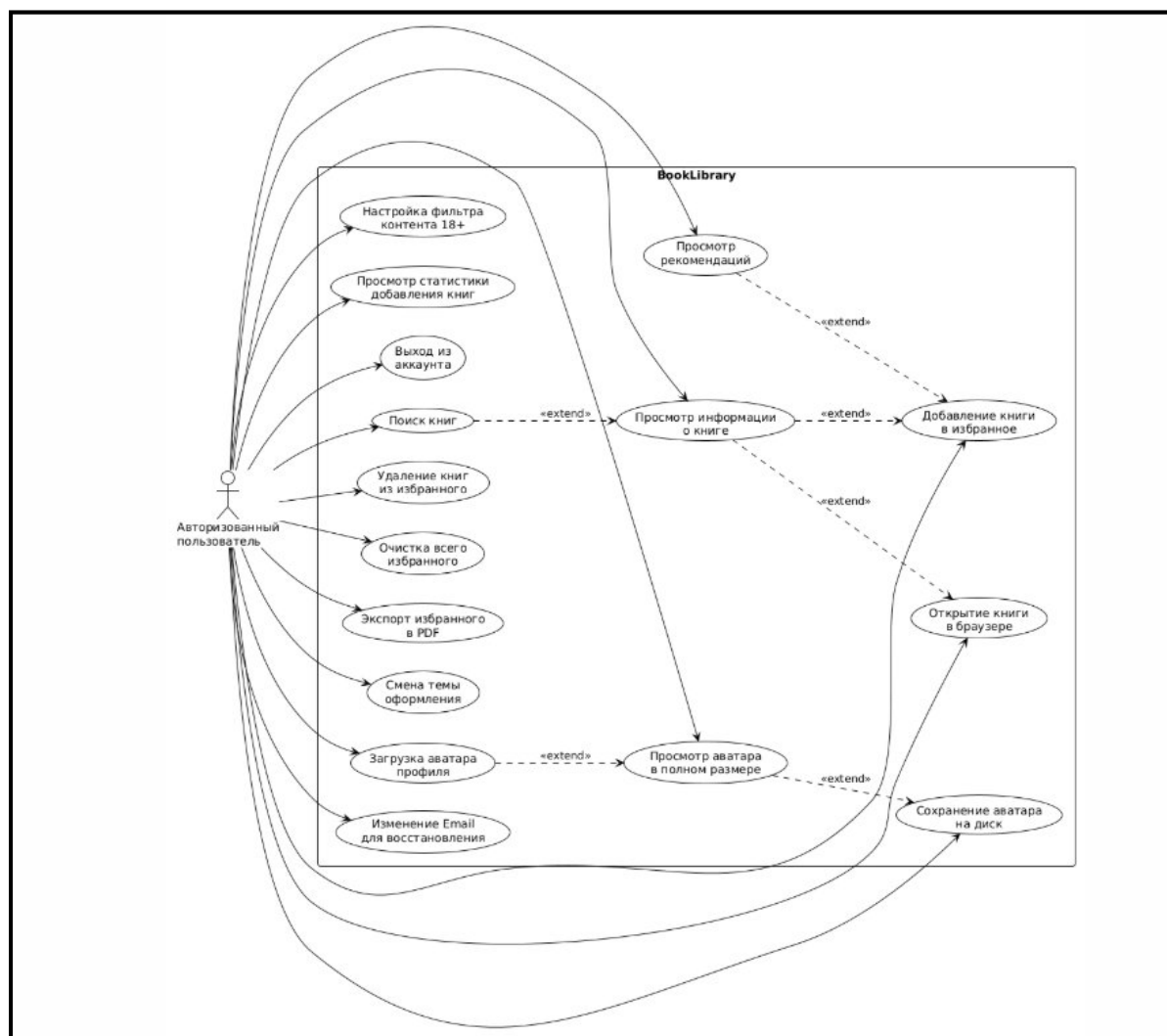


Рисунок 5 – Диаграмма прецедентов авторизованного пользователя

1.3.1 Требования к операционной системе

Приложение разработано на платформе .NET Framework 4.8 с использованием технологии Windows Forms, что определяет его целевую ориентацию на операционные системы семейства Microsoft Windows. Поддерживаются все актуальные версии данной ОС: Windows 10 (начиная с версии 1809), Windows 11 (все редакции), а также серверные версии Windows Server 2016, 2019 и 2022. Возможность запуска приложения в среде операционных систем Linux или macOS с использованием прослойки совместимости Wine теоретически существует, однако данный режим работы

не являлся целевым при проектировании, вследствие чего корректное функционирование всех компонентов не гарантируется.

1.3.2 Требования к процессору

Приложение не выполняет ресурсоёмких вычислительных операций, таких как обработка видео или сложные математические расчёты, поэтому требования к центральному процессору являются минимальными. Для работы приложения достаточно процессора архитектуры x86 или x64 с тактовой частотой не менее 1,6 ГГц. Рекомендуется использование двухъядерного процессора с частотой 2,4 ГГц и выше, что обеспечивает более отзывчивую работу пользовательского интерфейса при одновременном выполнении нескольких задач (например, асинхронная загрузка обложек во время прокрутки списка результатов поиска).

1.3.3 Требования к оперативной памяти

Базовое потребление оперативной памяти приложением в состоянии покоя составляет около 40–60 МБ, что обусловлено загрузкой необходимых библиотек .NET Framework и инициализацией графического интерфейса. При активной работе, включающей загрузку и кэширование обложек книг, потребление памяти может возрастать до 150–200 МБ. Исходя из этого, минимальным требованием является наличие 2 ГБ оперативной памяти, что соответствует спецификациям большинства современных офисных и домашних компьютеров начального уровня. Для комфортной работы, особенно при длительных сеансах с накоплением большого объёма, сохранённых в памяти изображений, рекомендуется объём ОЗУ от 4 ГБ и выше.

1.3.4 Требования к дисковому пространству

Установочный комплект приложения занимает порядка 15–20 МБ дискового пространства. В процессе эксплуатации приложение создаёт и пополняет локальную базу данных, а также сохраняет загруженные обложки книг в кэш – хранилище. Размер базы данных линейно зависит от количества

сохранённых в избранное книг и в среднем составляет 10–20 КБ на одну запись. Объём кэша обложек определяется пользовательской активностью: одно изображение среднего качества занимает около 30–50 КБ. Таким образом, для полноценной работы приложения с учётом хранения до 200 книг в избранном и соответствующего объёма кэшированных обложек рекомендуется наличие не менее 500 МБ свободного дискового пространства. Минимальное требование составляет 100 МБ без учёта кэша обложек.

1.3.5 Требования к разрешению экрана

Графический интерфейс приложения спроектирован с учётом адаптивной вёрстки и поддерживает изменение размеров окна перетаскиванием краёв. Тем не менее, для корректного отображения всех элементов управления без наложений и обрезания контента минимальное разрешение экрана должно составлять 1024×768 пикселей. Рекомендуемым является разрешение от 1366×768 пикселей и выше, что соответствует стандартным параметрам дисплеев современных ноутбуков и мониторов. При меньших разрешениях возможно возникновение вертикальной и горизонтальной полос прокрутки, что не влияет на функциональность, но может снижать удобство использования.

1.3.6 Требования к сетевому подключению

Функциональность поиска книг и загрузки обложек реализована через взаимодействие с публичным API Open Library по протоколу HTTPS. Для выполнения данных операций требуется активное широкополосное подключение к сети Интернет. Приложение использует асинхронные HTTP – запросы с таймаутом в 30 секунд, что позволяет работать даже на относительно медленных каналах связи. Рекомендуемая скорость соединения составляет от 10 Мбит/с для комфортной загрузки обложек без заметных задержек. В автономном режиме (при отсутствии подключения к Интернету) приложение сохраняет ограниченную функциональность: доступен просмотр ранее сохранённого списка избранного, локальной статистики, кэшированных

обложек и экспорт данных в PDF. Функции поиска новых книг и обновления рекомендаций в офлайн – режиме недоступны.

1.4 Выбор архитектуры системы

После того как были определены цели разрабатываемого приложения, необходимо определить его вид. Удобнее всего с поставленными задачами будет работать на основе десктопного приложения с использованием монолитной архитектуры. Монолитная архитектура представляет собой единый исполняемый модуль, который может работать автономно и независимо от других приложений или внешних веб – сервисов (за исключением обращения к открытому API для получения данных). Благодаря такому выбору архитектуры пользователю не нужно будет заморачиваться с установкой дополнительных серверов или настройкой сложного окружения, а для начала работы необходимо будет просто запустить наш разрабатываемый программный комплекс. Также данная архитектура имеет ряд удобных для нас преимуществ: упрощенное развертывание, высокая скорость работы, так как все компоненты находятся в одном процессе, и удобство отладки взаимодействия между модулями.

Для наглядной демонстрации на рисунке 6 показана диаграмма развёртывания.

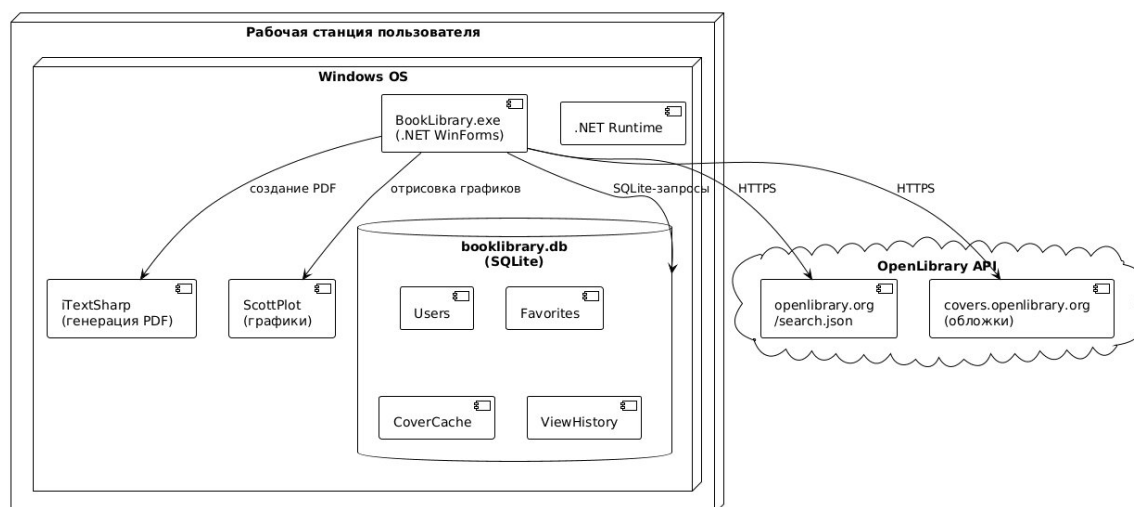


Рисунок 6 – Диаграмма развёртывания

Также для понимания алгоритма функционирования программы выполнено описание диаграммы последовательностей. Указанная диаграмма приведена на рисунке 7.

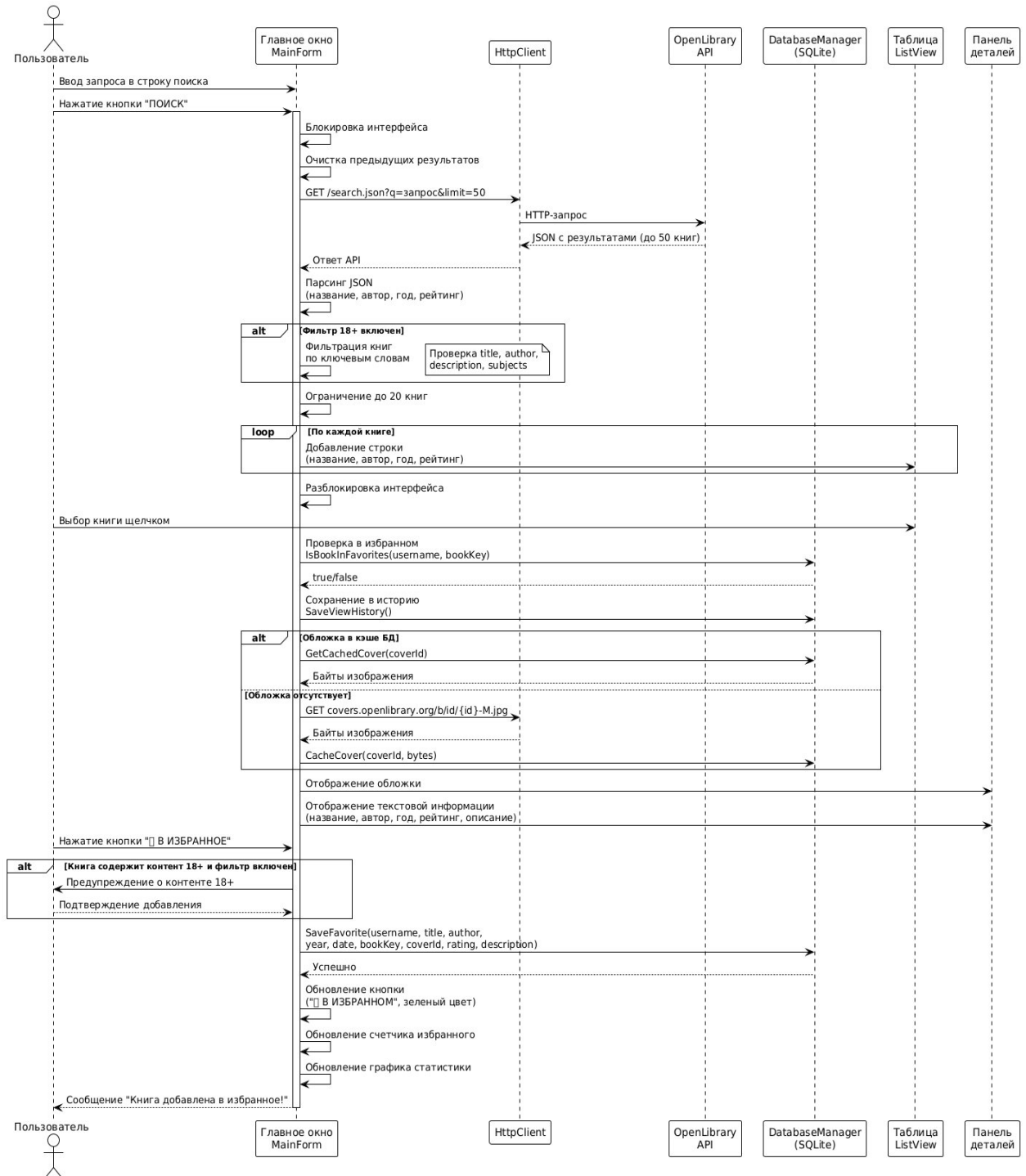


Рисунок 7 – Диаграмма последовательностей

Исходя из требуемого функционала, планируемой архитектуры проекта и выбранной целевой платформы был определен следующий набор инструментов разработки: язык программирования C#, среда разработки Microsoft Visual Studio 2022, ОС Microsoft Windows 10.

1.5 Выбор языка программирования

Для реализации данной задачи был выбран язык программирования C#. Данный язык является одним из наиболее популярных и широко используемых в современной разработке программного обеспечения. Он был разработан компанией Microsoft и входит в состав платформы Microsoft .NET, которая предоставляет широкий набор инструментов и библиотек для создания различных типов приложений.

Одним из ключевых преимуществ языка C# является его объектно-ориентированная модель программирования. Она позволяет структурировать программу в виде классов и объектов, что облегчает разработку, тестирование и дальнейшее сопровождение программного кода. Благодаря этому программы становятся более понятными, масштабируемыми и удобными для модификации[2].

Также важным фактором выбора C# является простота синтаксиса и высокая читаемость кода. Язык сочетает в себе элементы таких языков, как C++ и Java, но при этом предлагает более удобные и безопасные механизмы работы с памятью и обработкой ошибок.

Еще одним преимуществом является развитая экосистема разработки. Для написания программ на C# широко используется среда разработки Microsoft Visual Studio, которая предоставляет инструменты для написания, отладки и тестирования кода. Наличие большого количества готовых библиотек и фреймворков значительно ускоряет процесс разработки[2].

Кроме того, C# поддерживает создание различных типов приложений: настольных, веб – приложений, сервисов и мобильных программ. Это делает

язык универсальным инструментом, подходящим для реализации учебных и практических проектов.

1.6 Выбор среды разработки

В качестве среды разработки, исходя из основной платформы проекта, была взята Microsoft Visual Studio 2022.

Интегрированная среда разработки Visual Studio 2022 представляет собой современную многофункциональную платформу, обеспечивающую полный цикл жизненного цикла создания программного обеспечения — от проектирования архитектуры и написания исходного кода до комплексного тестирования и развертывания готовых решений[3]. Являясь первым в истории линейки полностью 64 – битным приложением, данная IDE снимает ограничения на объем используемой оперативной памяти, что гарантирует стабильную работу с крупными программными комплексами, ускоряет процессы индексации файлов и компиляции ресурсоемких проектов[3]. Среда поддерживает широкий спектр языков программирования, включая C#, C++, F# и Python, и предоставляет разработчику доступ к мощной экосистеме .NET, позволяющей создавать кроссплатформенные приложения с высокой производительностью. Ключевой особенностью Visual Studio 2022 является интеграция интеллектуальной системы автодополнения кода IntelliCode, которая на основе алгоритмов машинного обучения анализирует контекст разработки и предлагает оптимальные варианты завершения строк, минимизируя количество синтаксических ошибок[3]. Встроенный инструментарий отладки и диагностики включает в себя механизмы профилирования памяти и нагрузки на центральный процессор, а также инновационную функцию Hot Reload, позволяющую изменять логику программы и графический интерфейс в режиме реального времени без прерывания сеанса выполнения. Кроме того, среда обладает глубокой интеграцией с системами контроля версий Git и менеджером пакетов NuGet,

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		23

что упрощает управление внешними зависимостями и организацию совместной работы над кодом. Наличие визуальных редакторов для создания пользовательских интерфейсов (WPF, WinForms) и инструментов для прямого взаимодействия с базами данных позволяет вести разработку всех уровней приложения в рамках единого рабочего пространства, обеспечивая высокую скорость реализации проектных задач и соответствие современным стандартам индустрии программного обеспечения.

Данная среда программирования бесплатна и ее сайт представлен на рисунке 8.

Познакомьтесь с семейством продуктов Visual Studio

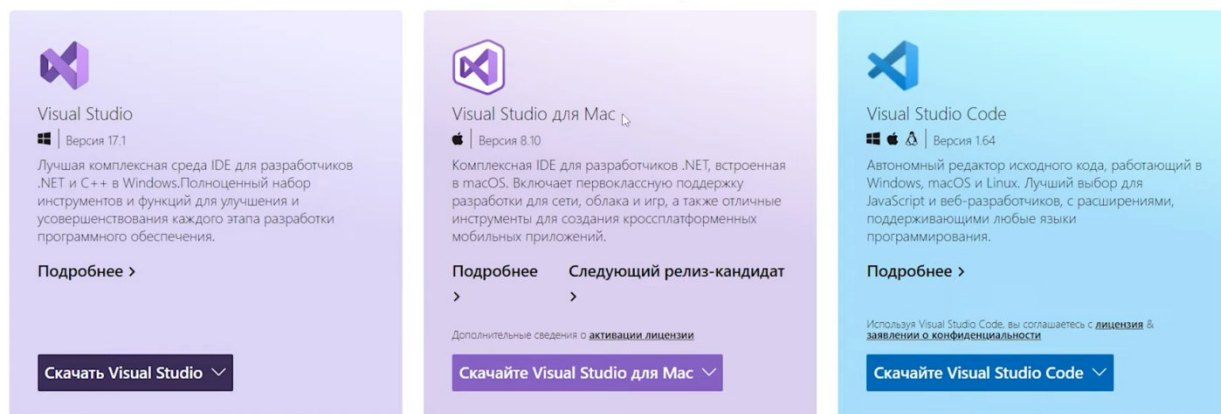


Рисунок 8—Сайт Microsoft Visual Studio

Страница загрузки Visual Studio включает:

- подробное описание функциональных возможностей среды;
- технические требования к системе;
- другие возможные версии;
- форму для обратной связи;
- кнопки загрузки установщика.

Для того чтобы скачать Visual Studio необходимо нажать на кнопку загрузки, что можно увидеть на рисунке 9.



Рисунок 9— Область загрузки среды программирования на сайте

После мы переходим на другую страницу, где происходит скачивание установщика Visual Studio, и после скачивания открываем его (данное окно показано на рисунке 10).

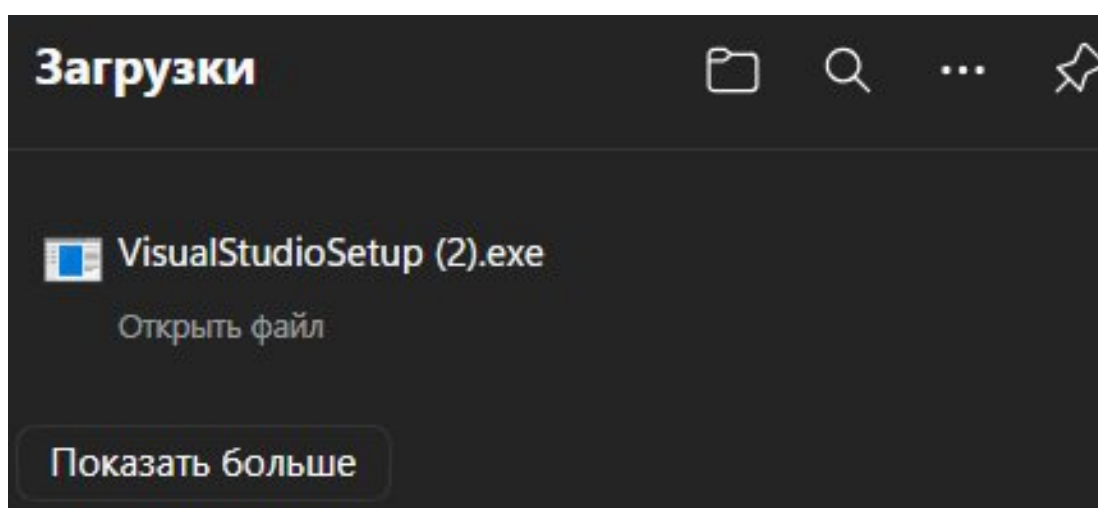


Рисунок 10— Загрузка установочного файла среды программирования

После открытия файла запускается процесс загрузки установки среды для выбора версии разработки. Ход загрузки отображается в интерфейсе и проиллюстрирован на рисунке 11.

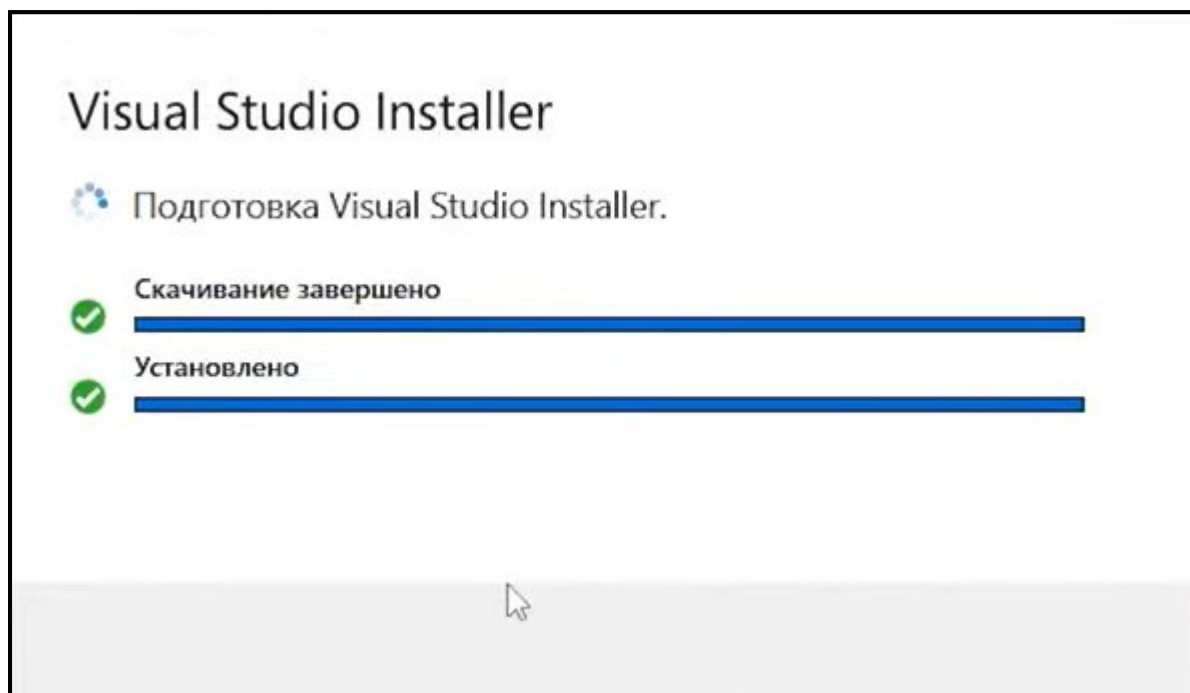


Рисунок 11— Окно установщика

После установки открывается окно с выбором нужных компонентов для версии Visual Studio 2022. Запуск файла инициирует работу Мастера установки — для продолжения процесса нужно нажать кнопку «Установить» в открывшемся окне (рисунок 12). Далее потребуется ознакомиться с лицензионным соглашением и подтвердить его принятие.

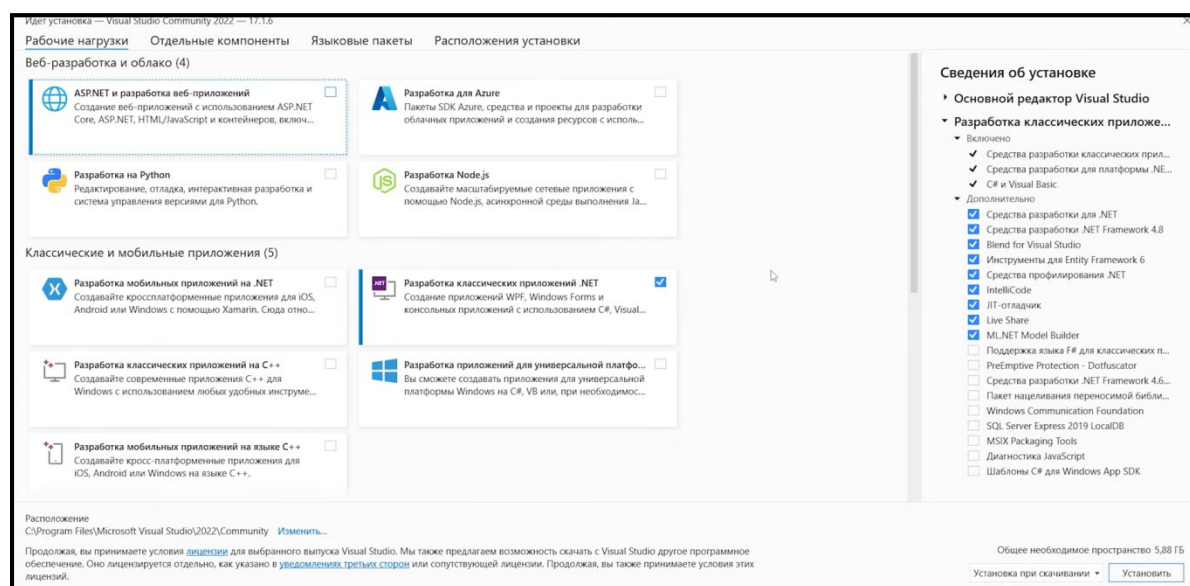


Рисунок 12— Мастер установки Visual Studio

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.11.001 ПЗ

Лист

26

После успешной установки мы видим кнопку «Запустить», которая представлена на рисунке 13.



Рисунок 13— Запуск среды программирования

Далее после запуска мы видим главное окно среды разработки (рисунок 14).

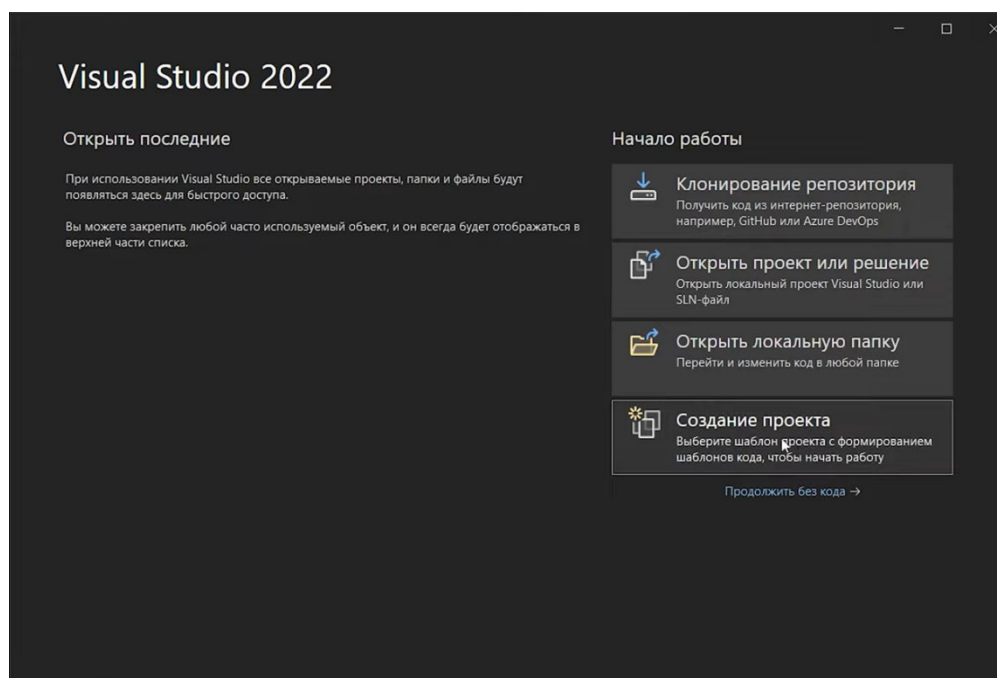


Рисунок 14— Главное окно среды программирования

После нажимаем на кнопку создания проекта (рисунок 15).

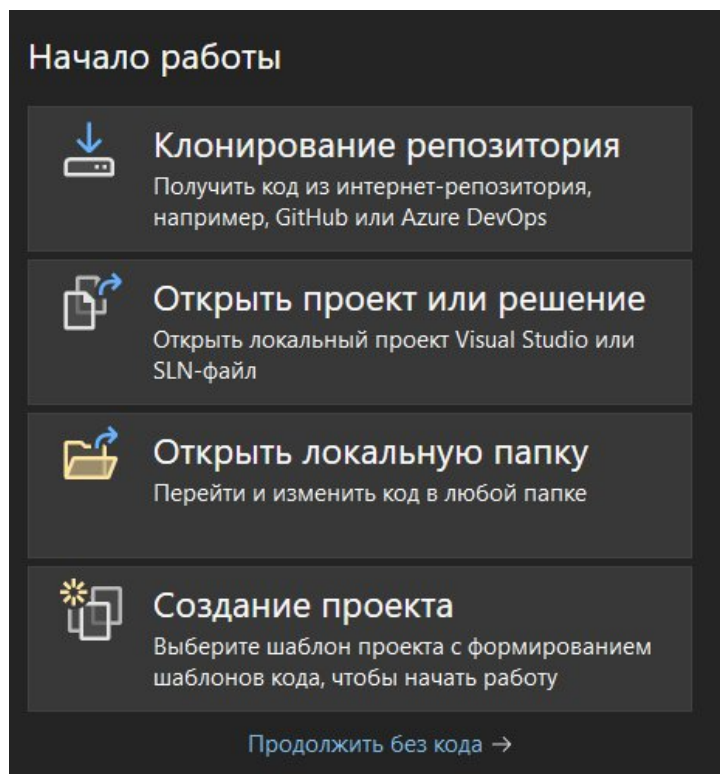


Рисунок 15— Создание нового проекта

Затем выбираем шаблон проекта, представленный на рисунке 16.

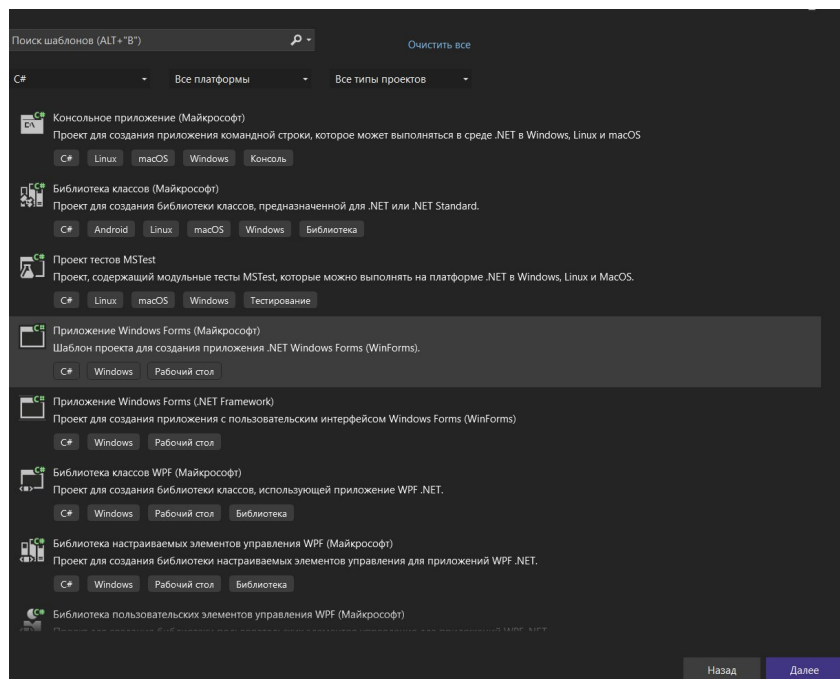


Рисунок 16— Выбор шаблона

Для создания проекта необходимо выбрать название, имя решения и папку, в которую будут сохраняться изменения по проекту, как представлено на рисунке 17.

Рисунок 17— Настройка параметров нового проекта

Затем нажимаем на кнопку «Далее», и на рисунке 18 показана целевая платформа, на которой будет основан проект.

Рисунок 18— Выбор целевой платформы проекта

Нажимаем на кнопку «Создать», и перед пользователем появляется окно разработки приложения, как это показано на рисунке 19. Можно заметить структуру решения в «Обозревателе решений», главный файл программного кода и дополнительные настройки среды разработки.

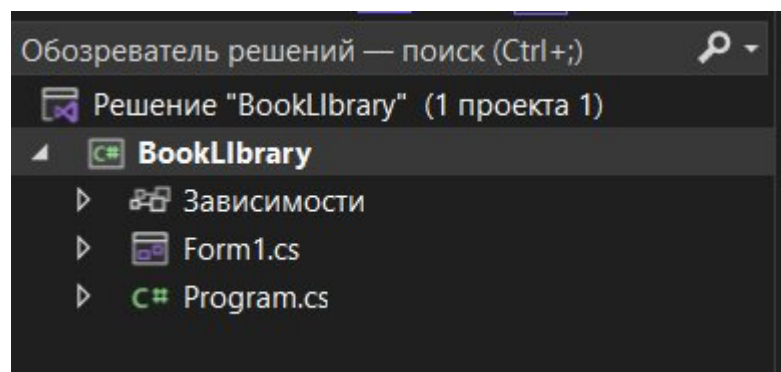


Рисунок 19— Структура созданного проекта

Для дальнейшей успешной разработки программы потребуется установить дополнительные библиотеки в диспетчере пакетов NuGet. Для этого выбираем вкладку «Проект» и переходим в раздел «Управление пакетами NuGet». Далее выбираем необходимые для разработки сторонние библиотеки (рисунок 20).

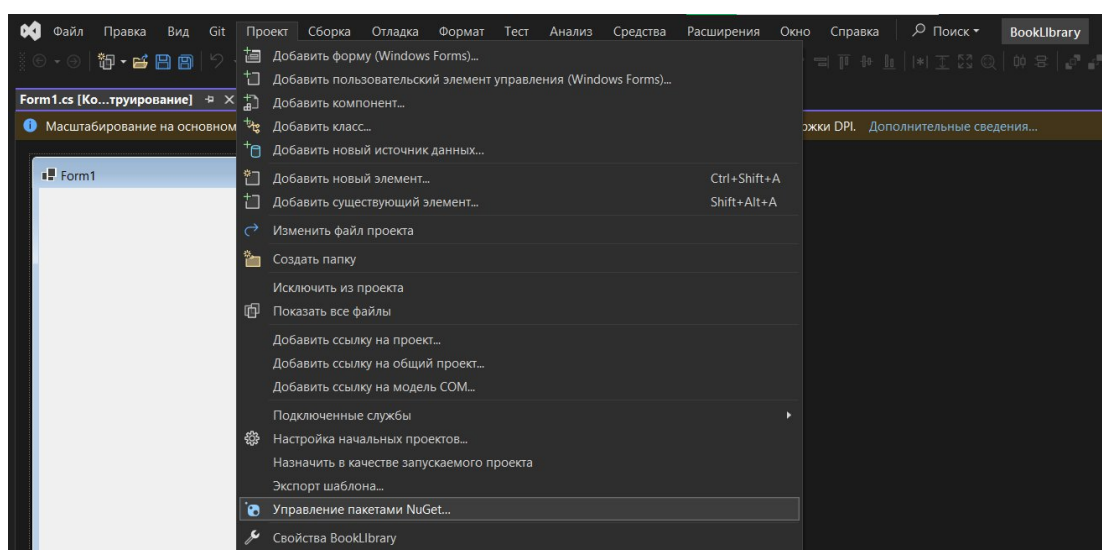


Рисунок 20— Диспетчер пакетов NuGet

Функциональность данных библиотек позволяет комфортно и быстро ускорить процесс разработки приложения, автоматизируя рутинные операции и позволяя сосредоточиться на бизнес – логике кода. Все дополнительные библиотеки показаны на рисунке 21.

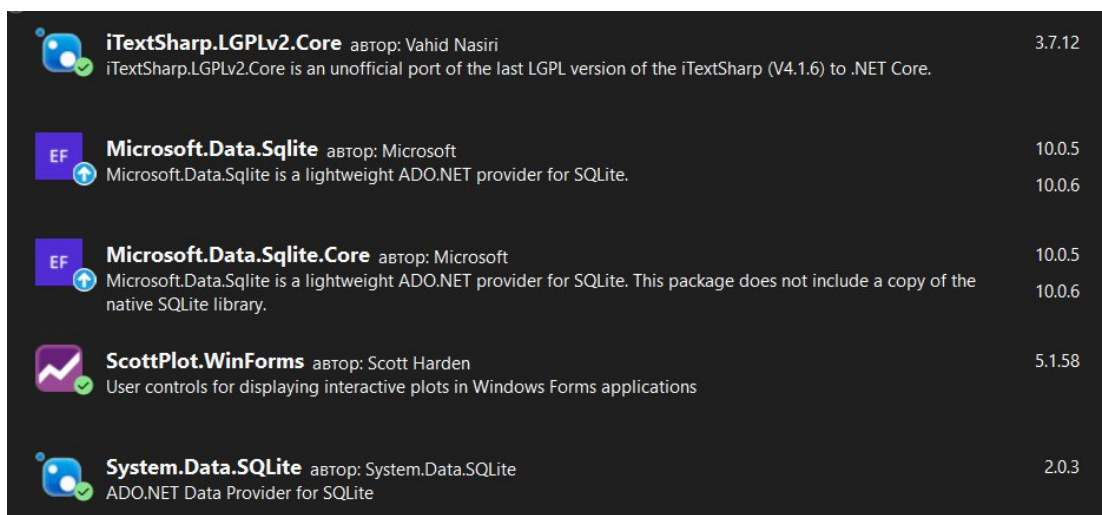


Рисунок 21— Установка дополнительных библиотек

После завершения процесса настройки среды программирования и подключения всех необходимых зависимостей был начат этап разработки управляющей программы.

1.7 Разработка управляющей программы

После завершения конфигурирования среды программирования началась стадия непосредственной разработки управляющей программы.

Программа построена по принципу монолитного клиентского приложения с трёхуровневой архитектурой: уровень представления, чем является графический интерфейс пользователя, уровень бизнес – логики, включающий в себя обработку запросов, фильтрацию контента, формирование рекомендаций и уровень доступа. Такое разделение обеспечивает модульность кода, упрощает его сопровождение и тестирование.

Пользовательский интерфейс реализован с использованием уникальных элементов управления. Для кнопок разработан собственный класс ModernButton, обеспечивающий скруглённые углы, плавные анимации при наведении и поддержку различных цветовых схем. Форма аутентификации и главное окно приложения поддерживают изменение

размеров с перехватом системных сообщений, что позволяет пользователю настраивать рабочее пространство по своему усмотрению.

Взаимодействие с внешним источником библиографических данных реализовано через публичный REST API OpenLibrary. Приложение отправляет асинхронные HTTP – запросы к первоисточнику, передавая поисковую строку и параметры фильтрации. В полученных JSON – ответах выделяется главная информация использованием встроенной библиотеки, после чего каждый объект преобразуется в экземпляр внутреннего класса Book, содержащего все необходимые метаданные: уникальный ключ, название, автора, год первой публикации, идентификатор обложки, средний рейтинг и краткую аннотацию.

Для оптимизации загрузки изображений обложек реализован двухуровневый механизм кэширования. При первом обращении к книге изображение обложки загружается по URL и сохраняется в таблице базы данных в виде BLOB – объекта. При повторном отображении той же книги загрузка происходит непосредственно из локального кэша, что значительно сокращает время отклика и снижает нагрузку на сетевое соединение.

На рисунке 22 представлена диаграмма классов проекта.

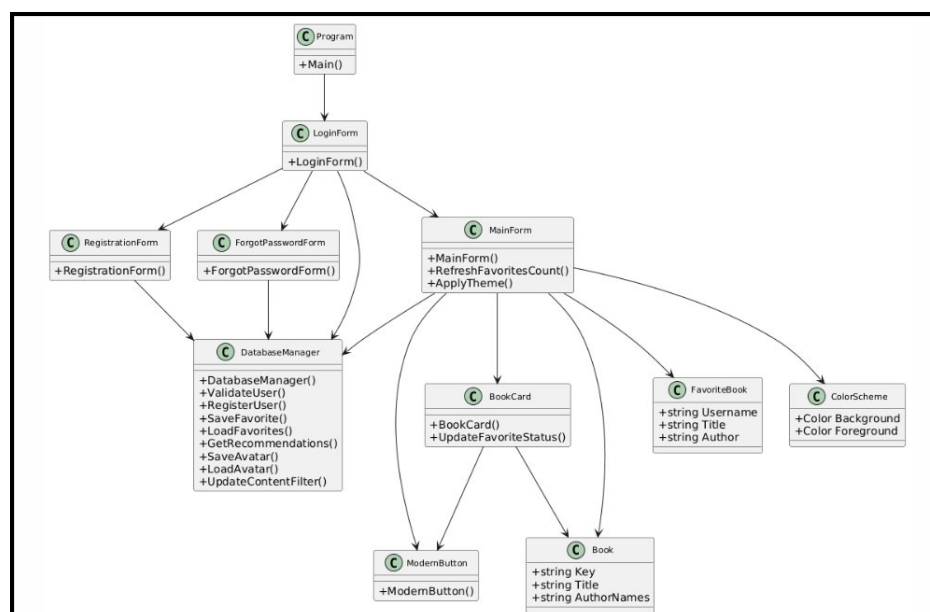


Рисунок 22—Диаграмма классов

Функциональная схема выполнения программы показана на рисунке 23.

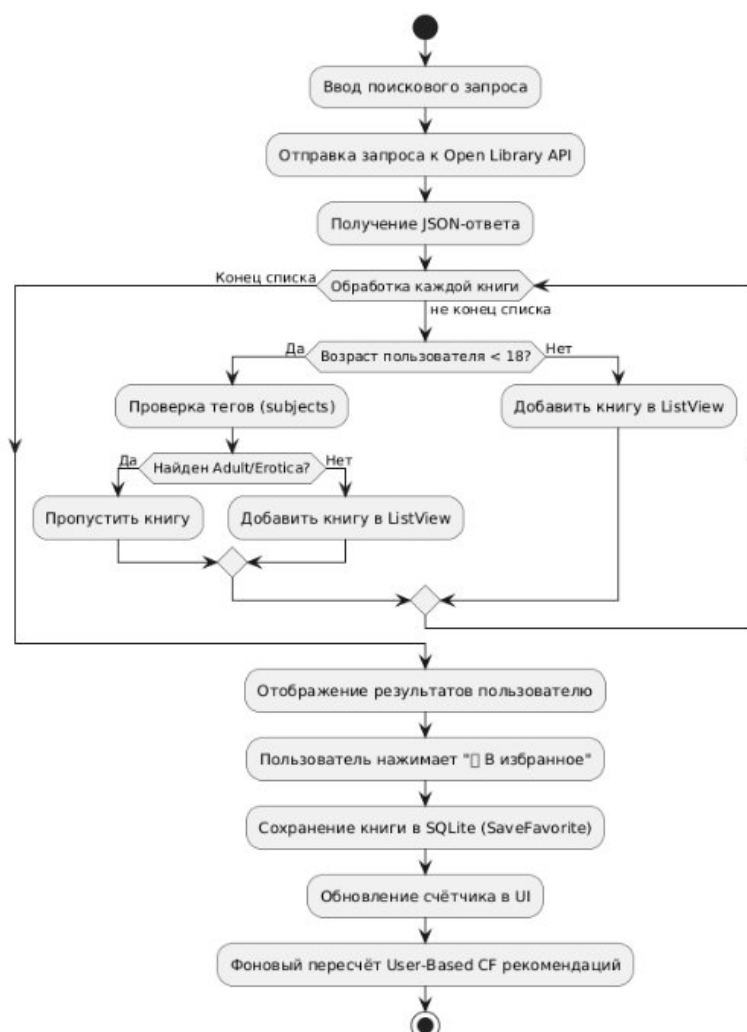


Рисунок 23— Функциональная схема работы проекта

В качестве системы управления базой данных была выбрана SQLite — легковесная встраиваемая реляционная СУБД, не требующая развертывания отдельного сервера и полностью хранящая данные в одном файле. Данный выбор обусловлен архитектурой приложения как настольного клиента, ориентированного на персональное использование: такой подход обеспечивает нулевую конфигурацию при установке, портативность данных и соблюдение принципов ACID при выполнении транзакций.

Структура базы данных включает четыре таблицы, покрывающие все функциональные потребности приложения. Центральной является

таблица Users, которая помимо учетных данных (логин, пароль, email для восстановления доступа) хранит индивидуальные настройки интерфейса каждого пользователя — тему оформления, возраст и флаг включения фильтра запрещенного контента, а также аватар профиля в бинарном формате BLOB. Таблица Favorites реализует связь «одинкомногим» между пользователем и сохраненными книгами, фиксируя не только ключ и идентификатор обложки из API OpenLibrary, но и полные метаданные издания: название, автора, год публикации, рейтинг и текстовое описание. Уникальное составное ограничение (Username, BookKey) гарантирует целостность данных на уровне СУБД, исключая дублирование записей. Вспомогательная таблица CoverCache реализует механизм локального кеширования изображений обложек, сохраняя бинарные данные в столбце типа BLOB вместе с датой помещения в кеш, что позволяет существенно сократить число повторных HTTP – запросов к внешнему сервису и ускорить отрисовку графического интерфейса. Таблица ViewHistory ведет журнал просмотров страниц книг пользователем, фиксируя временные метки каждого обращения — эти данные служат основой для будущего развития рекомендательных алгоритмов на основе коллаборативной фильтрации.

Обновление схемы реализовано по принципу «миграций на лету»: при инициализации приложение проверяет наличие требуемых столбцов с помощью команд PRAGMA table_info, автоматически применяя операторы ALTER TABLE ADD COLUMN при обнаружении несоответствий, что позволяет безопасно расширять функциональность без потери уже накопленных пользовательских данных. Все операции с базой выполняются исключительно через параметризованные SQL – запросы, что полностью исключает риск SQL–инъекций и обеспечивает корректную обработку специальных символов в пользовательском вводе. На рисунке 24 есть возможность ознакомления с диаграммой базы данных приложения.

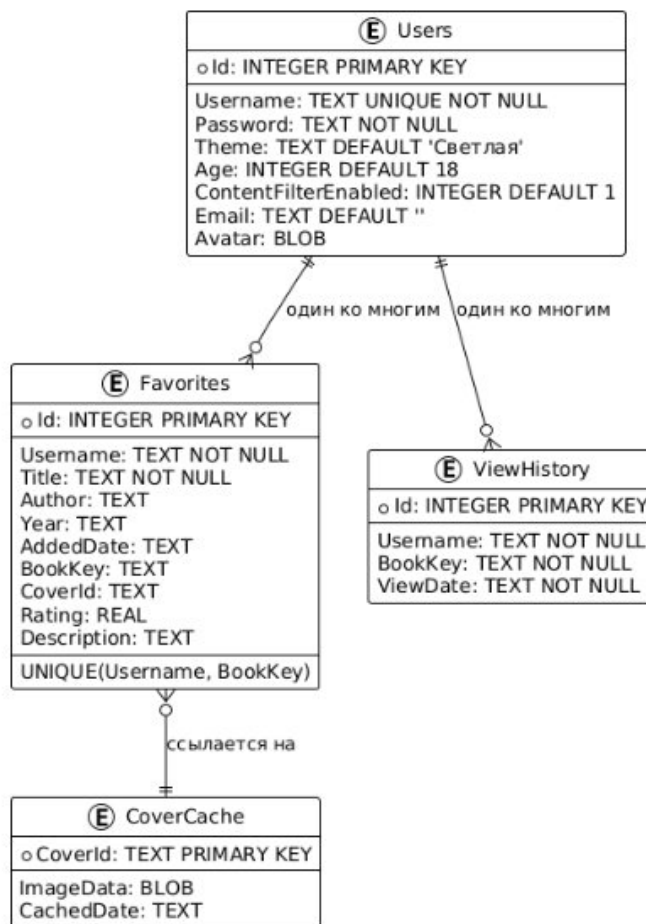


Рисунок 24– Диаграмма базы данных

1.8 Графический интерфейс программы

1.8.1 Окно аутентификации пользователя

При запуске приложения пользователь попадает на форму аутентификации, выполняющую функции входа в систему, регистрации нового аккаунта и восстановления утраченного пароля. Внешний вид окна аутентификации приведен на рисунке 25.

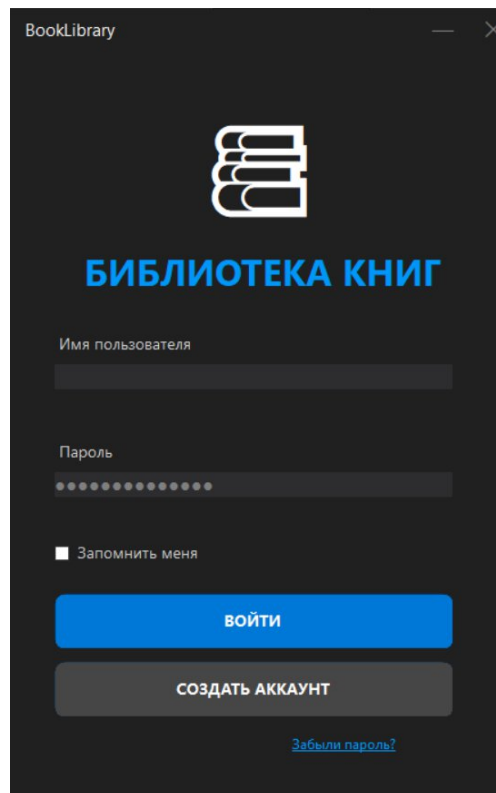


Рисунок 25 — Окно аутентификации пользователя

Форма реализована с использованием кастомного оформления: стандартные элементы управления WinForms заменены на стилизованные текстовые поля с подчеркиванием цвета акцента и кнопки со скругленными углами, наследуемые от класса ModernButton. Перемещение окна реализовано через перехват событий мыши на панели заголовка, а изменение размеров — через обработку системного сообщения WM_NCHITTEST с вычислением координат курсора для определения границ. Логика проверки учетных данных основана на методе ValidateUser класса DatabaseManager, выполняющем поиск совпадения пары «логин–пароль» с одновременным извлечением пользовательских настроек — темы оформления, возраста и статуса фильтра контента. Фрагмент кода валидации представлен в листинге 1.

```
if (dbManager.ValidateUser(username, password, out string
theme, out int age, out bool contentFilterEnabled))
{
    Hide();
}
```

```

        var mainForm = new MainForm(username, theme, age,
contentFilterEnabled, dbManager);

        mainForm.FormClosed += (s, args) => Close();
        mainForm.Show();
    }

    else    MessageBox.Show("Неверный логин или пароль!",    "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);

```

Листинг 1 — Проверка учетных данных пользователя

1.8.2 Окно регистрации нового пользователя

При нажатии кнопки «Создать аккаунт» на форме входа открывается окно завершения регистрации, где запрашивается дополнительная информация: возраст пользователя, согласие на включение контентного фильтра и адрес электронной почты для восстановления пароля. Окно регистрации показано на рисунке 26.

Рисунок 26 — Окно регистрации с дополнительными настройками

Ключевой особенностью данной формы является принудительное включение фильтра контента для пользователей младше 18 лет. При изменении значения числового поля возраста динамически обновляется

информационная метка и состояние флажка фильтра: если возраст меньше 18, флажок принудительно устанавливается в активное положение и блокируется от изменения. Программная реализация данного ограничения приведена в листинге 2.

```

ageNumeric.ValueChanged += (s, e) => {
    bool isAdult = ageNumeric.Value >= 18;
    if (!isAdult) {
        contentFilterCheckBox.Checked = true;
        contentFilterCheckBox.Enabled = false;
        ageInfoLabel.Text = "⚠ Вам меньше 18 лет. Фильтр
контента обязателен.";
        ageInfoLabel.ForeColor = DrawingColor.Orange;
    } else {
        contentFilterCheckBox.Enabled = true;
        ageInfoLabel.Text = "✓ Вам 18 лет или больше. Вы можете
управлять фильтром в профиле.";
        ageInfoLabel.ForeColor = DrawingColor.Green;
    }
};

```

Листинг 2 — Логика возрастных ограничений при регистрации

После заполнения всех полей вызывается метод RegisterUser класса DatabaseManager, выполняющий SQL-запрос на вставку новой записи в таблицу Users с проверкой уникальности логина через встроенный механизм ограничений SQLite.

1.8.3 Окно восстановления пароля

Для случаев утери пароля реализована форма восстановления доступа, открываемая по ссылке «Забыли пароль?» с главного экрана аутентификации. Пользователю предлагается ввести адрес электронной почты, указанный при регистрации, после чего система выполняет поиск соответствующей учетной записи и генерирует новый пароль. Сгенерированный пароль сохраняется в текстовый файл во временной директории системы и автоматически

открывается стандартным текстовым редактором. Обновление пароля в базе данных выполняется методом UpdatePassword, фрагмент которого представлен в листинге 3.

Листинг 3 — Обновление пароля пользователя в базе данных

```
public void UpdatePassword(string username, string
newPassword)
{
    using (var cmd = new SQLiteCommand("UPDATE Users SET
Password = @p WHERE Username = @u;", _connection))
    {
        cmd.Parameters.AddWithValue("@p", newPassword);
        cmd.Parameters.AddWithValue("@u", username);
        cmd.ExecuteNonQuery();
    }
}
```

Листинг 3 — Обновление пароля пользователя в базе данных

1.8.4 Вкладка поиска в главном окне приложения

После успешной аутентификации пользователь попадает в главное окно приложения, организованное в виде многостраничного интерфейса с вкладками. Первая вкладка «Поиск» предоставляет функциональность поиска книг через публичный API OpenLibrary. Интерфейс вкладки разделен на три зоны: панель управления поиском, таблица результатов и область детальной информации. Схема взаимодействия компонентов вкладки поиска представлена на рисунке 27.

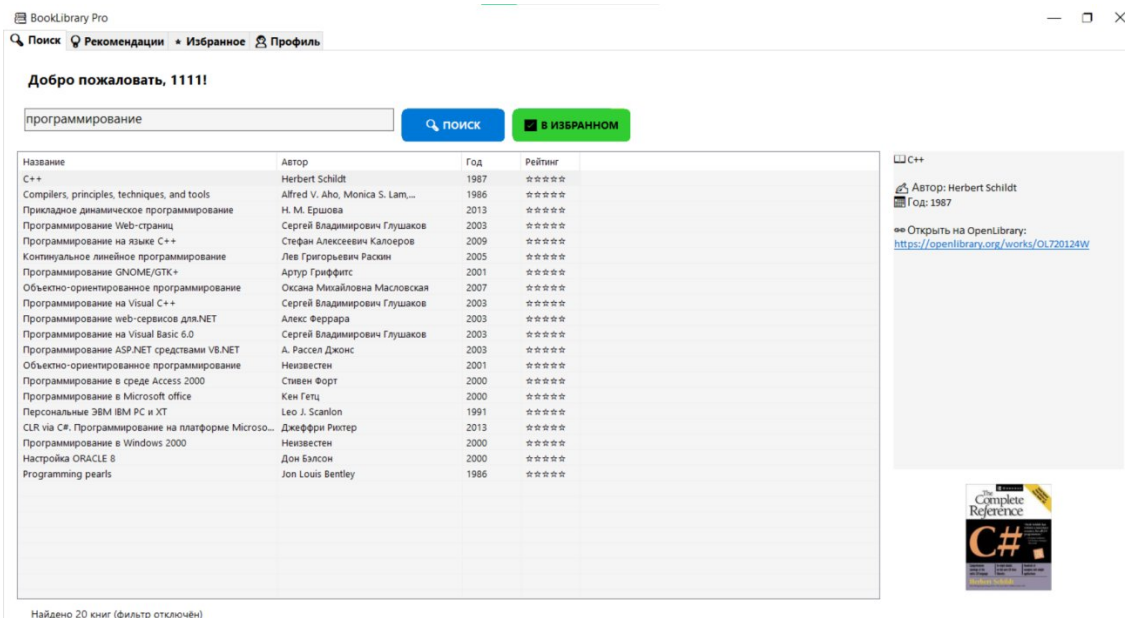


Рисунок 27 — Вкладка поиска книг с загруженными результатами

Поисковый запрос формируется путем отправки GET-запроса к эндпоинту `https://openlibrary.org/search.json` с передачей поисковой строки и лимита результатов. Ответ API парсится с использованием библиотеки `System.Text.Json`, и каждый JSON-объект преобразуется в экземпляр класса `Book`. Фрагмент кода выполнения HTTP-запроса и парсинга представлен в листинге 4.

```
private async Task<List<Book>> SearchOpenLibrary(string
query, int limit)
{
    var books = new List<Book>();

    string url =
$"{OPEN_LIBRARY_SEARCH_URL}?q={Uri.EscapeDataString(query)}&limit
={limit}";

    var response = await httpClient.GetAsync(url);
    if (!response.IsSuccessStatusCode) return books;
    string json = await response.Content.ReadAsStringAsync();
    using JsonDocument doc = JsonDocument.Parse(json);
    if (doc.RootElement.TryGetProperty("docs", out
JsonElement docs))
```

```

        foreach (var el in docs.EnumerateArray())
        {
            var book = ParseBookFromJson(el);
            if (book != null
&& !string.IsNullOrEmpty(book.Title)) books.Add(book);
        }
    return books;
}

```

Листинг 4 — Поиск книг через OpenLibrary API

Отдельного внимания заслуживает механизм фильтрации контента 18+. Класс MainForm содержит статический набор ключевых слов AdultKeywords на русском и английском языках, охватывающий различные формы обозначения материалов для взрослых. Метод IsBookSafe последовательно проверяет наличие запрещенных ключевых слов в названии книги, имени автора, описании и списке тематик. Если контентный фильтр активен, неприемлемые книги исключаются из результирующей выборки, а в строке статуса отображается количество скрытых результатов.

При выборе книги в таблице результатов асинхронно загружается изображение обложки с предварительной проверкой локального кеша. Описание книги, включая автора, год издания, рейтинг в виде звездного индикатора и аннотацию, отображается в многострочном текстовом поле с поддержкой активных гиперссылок, позволяющих открыть страницу книги в браузере напрямую из приложения.

1.8.5 Вкладка рекомендаций

Вкладка рекомендаций реализует упрощенный алгоритм коллаборативной фильтрации для формирования персональных книжных предложений. Внешний вид вкладки показан на рисунке 28.

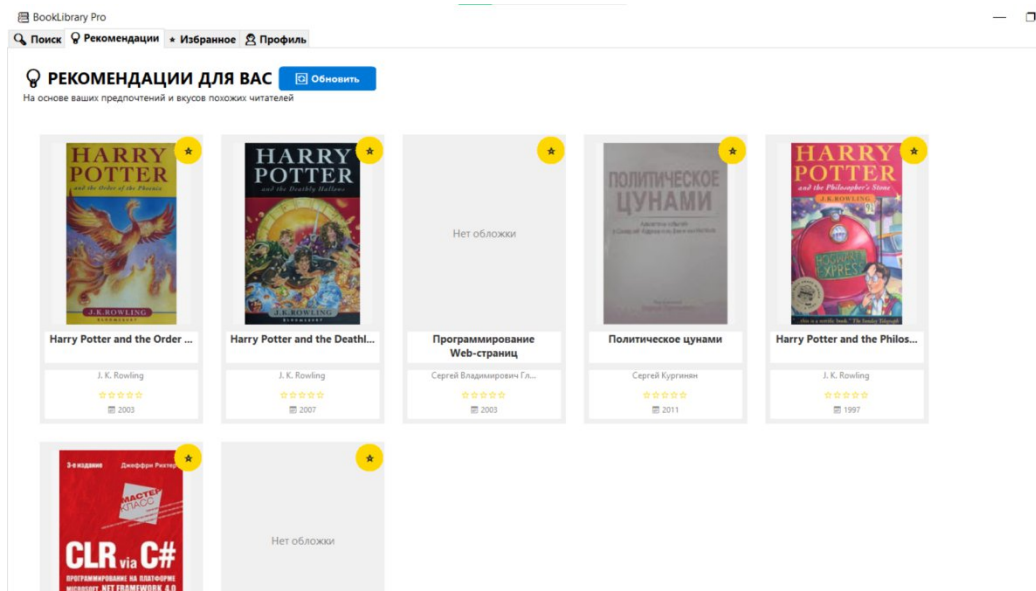


Рисунок 28 — Вкладка рекомендаций с карточками книг

Алгоритм генерации рекомендаций работает по следующему принципу: для текущего пользователя извлекается множество уже добавленных в избранное книг; затем в таблице Favorites выполняется поиск других пользователей, имеющих максимальное пересечение по составу избранного с текущим; из книг этих «похожих» читателей выбираются те, которых у текущего пользователя еще нет в коллекции. При отсутствии достаточного количества данных (менее трех книг в избранном) пользователю выводится информационное сообщение с предложением пополнить коллекцию. Фрагмент алгоритма формирования рекомендаций приведен в листинге 5.

```
varsimilarUsers = newDictionary<string, int>();
var inClause = string.Join(",", userBooks.Select((_, i) =>
$"@k{i}"));
using (var cmd = new SQLiteCommand($"SELECT Username,
COUNT(*) as CommonCount FROM Favorites WHERE BookKey IN
({inClause}) AND Username != @u GROUP BY Username HAVING
CommonCount >= 1 ORDER BY CommonCount DESC LIMIT 10;",
_connection))
{
    cmd.Parameters.AddWithValue("@u", username);
```

```

        for (int i = 0; i < userBooks.Count; i++)
            cmd.Parameters.AddWithValue($"@k{i}", userBooks.ElementAt(i));
        using (var reader = cmd.ExecuteReader())
            while (reader.Read())
                similarUsers[reader.GetString(0)] =
reader.GetInt32(1);
    }

```

Листинг 5 — Поиск похожих пользователей для генерации рекомендаций

Результаты рекомендаций отображаются в виде плиточного интерфейса с использованием компонента `FlowLayoutPanel`, где каждая книга представлена карточкой класса `BookCard`, содержащей обложку, название, автора, звездный рейтинг и кнопку добавления в избранное. Для каждой карточки реализованы визуальные эффекты наведения с подсветкой границы и индикацией статуса «уже в избранном» через изменение цвета и текста кнопки.

1.8.6 Вкладка избранного

Вкладка избранного предоставляет функциональность управления сохраненной пользовательской коллекцией книг. Интерфейс включает таблицу со списком изданий и панель управления с тремя операциями: выборочное удаление отмеченных записей, полная очистка коллекции и экспорт списка в формат PDF. Фрагмент кода экспорта данных в PDF–документ представлен в листинге 6.

```

document.Open();
var titleFont = new iTextSharp.text.Font(baseFont, 18,
iTextSharp.text.Font.BOLD);
var title = new Paragraph($"Список избранных книг пользователя
{currentUsername}", titleFont);
title.Alignment = Element.ALIGN_CENTER;
document.Add(title);
var table = new PdfPTable(4);
table.WidthPercentage = 100;

```

```

        table.AddCell(new PdfPCell(new Phrase("Название",
headerFont)));
        table.AddCell(new PdfPCell(new Phrase("Автор", headerFont)));
        table.AddCell(new PdfPCell(new Phrase("Год", headerFont)));
        table.AddCell(new PdfPCell(new Phrase("Добавлено",
headerFont)));
        foreach (var book in favorites)
        {
            table.AddCell(new PdfPCell(new Phrase(book.Title ?? "",
cellFont)));
            table.AddCell(new PdfPCell(new Phrase(book.Author ?? "",
cellFont)));
            table.AddCell(new PdfPCell(new Phrase(book.Year ?? "",
cellFont)));
            table.AddCell(new PdfPCell(new Phrase(book.AddedDate ??
"", cellFont)));
        }
        document.Add(table);
        document.Close();

```

Листинг 6 — Формирование PDF-отчета по избранным книгам

Для экспорта используется библиотека iTextSharp, формирующая структурированный документ с заголовком, таблицей избранных книг и информационной строкой с датой создания отчета. Для корректного отображения кириллических символов реализовано автоматическое определение доступного шрифта: при наличии файла arial.ttf используется он, иначе применяется встроенный шрифт Helvetica с поддержкой Unicode-кодировки.

1.8.7 Вкладка профиля и статистики

Вкладка профиля объединяет функции управления учетной записью и визуализации статистики чтения. Пользовательский интерфейс разделен на три функциональные зоны: информационная панель с аватаром и текущими настройками, панель управления темой оформления и контентным фильтром,

а также область графика статистики. Внешний вид вкладки профиля показан на рисунке 29.



Рисунок 29 — Вкладка профиля с графиком статистики

Аватар пользователя хранится в базе данных в бинарном формате. При клике на изображение открывается полноэкранный просмотр с возможностью масштабирования и сохранения оригинала в файл. Загрузка нового аватара выполняется через стандартный диалог выбора файла с фильтрацией по допустимым форматам изображений. Перед сохранением изображение масштабируется методом высококачественной бикубической интерполяции для оптимизации размера хранимых данных.

График статистики строится с использованием библиотеки ScottPlot на основе данных о датах добавления книг в избранное. Метод RefreshStatsPlot извлекает из базы данных сгруппированную по датам статистику и формирует столбчатую диаграмму с автоматической настройкой отображения подписей в зависимости от количества записей: при малом числе дат подписи выводятся горизонтально стандартным шрифтом, при среднем — с наклоном 45 градусов, а при большом количестве записей применяется прореживание с шагом и вертикальная ориентация для предотвращения

наложения текста. Фрагмент кода адаптивной настройки осей диаграммы представлен в листинге 7.

```

        if (dates.Count <= 7)
        {
            plotView.Plot.Axes.Bottom.SetTicks(positions,
dates.ToArray());

            plotView.Plot.Axes.Bottom.TickLabelStyle.Rotation = 0;
            plotView.Plot.Axes.Bottom.TickLabelStyle.FontSize = 12;
        }
        else if (dates.Count <= 15)
        {
            plotView.Plot.Axes.Bottom.SetTicks(positions,
dates.ToArray());

            plotView.Plot.Axes.Bottom.TickLabelStyle.Rotation = 45;
            plotView.Plot.Axes.Bottom.TickLabelStyle.FontSize = 10;
        }
        else
        {
            int step = dates.Count / 15;
            var filteredTicks = new List<double>();
            var filteredLabels = new List<string>();
            for (int i = 0; i < positions.Length; i += step)
            {
                filteredTicks.Add(positions[i]);
                filteredLabels.Add(dates[i]);
            }

            plotView.Plot.Axes.Bottom.SetTicks(filteredTicks.ToArray(),
filteredLabels.ToArray());

            plotView.Plot.Axes.Bottom.TickLabelStyle.Rotation = 90;
            plotView.Plot.Axes.Bottom.TickLabelStyle.FontSize = 8;
        }
    }

```

Листинг 7 — Адаптивная настройка подписей оси графика статистики

Смена темы оформления реализована через перечисление доступных цветовых схем, хранящихся в словаре themes класса MainForm. Каждая схема описывает цвета фона, текста, элементов управления и акцентные цвета. При выборе новой темы вызывается рекурсивный обход всех дочерних элементов управления формой с обновлением их цветовых свойств — фон панелей, цвет текста меток, фоновый цвет таблиц и полей ввода. Данный подход позволяет централизованно управлять оформлением без необходимости ручного обновления каждого элемента.

1.9 Тестирование и анализ результатов

Для проверки корректности работы программного обеспечения был составлен набор тестовых сценариев, охватывающих основные функциональные возможности системы. Тестирование проводилось методом ручного ввода с проверкой как штатных сценариев использования, так и обработки исключительных ситуаций.

1.9.1 План тестирования

В таблице 2 представлен перечень тестовых сценариев с указанием ожидаемых результатов.

Таблица 2 — Сценарии тестирования

Наименование сценария	Действия пользователя	Ожидаемый результат
Регистрация нового пользователя	Заполнение всех полей, нажатие «Завершить регистрацию»	Создание учетной записи, отображение сообщения об успехе
Регистрация с существующим логином	Ввод уже зарегистрированного логина	Отображение сообщения об ошибке
Наименование	Действия пользователя	Ожидаемый результат

сценария		
Аутентификация пользователя	Ввод корректных логина и пароля, нажатие «Войти»	Успешный переход в главное окно приложения
Аутентификация с неверными данными	Ввод неверного логина или пароля	Отображение сообщения «Неверный логин или пароль»
Поиск и фильтрация книг	Ввод поискового запроса, включение фильтра 18+	Отображение результатов без взрослого контента
Загрузка некорректного файла аватара	Выбор файла неподдерживаемого формата	Отображение сообщения об ошибке, аватар не изменяется
Добавление книги в избранное	Выбор книги в результатах поиска, нажатие «В избранное»	Сохранение книги в базу данных, уведомление об успехе
Повторное добавление книги	Попытка добавить уже существующую в избранном книгу	Отображение сообщения «Книга уже в избранном»
Генерация рекомендаций	Добавление 3+ книг в избранное, переход на вкладку рекомендаций	Отображение карточек с рекомендуемыми книгами
Наименование	Действия пользователя	Ожидаемый результат

сценария		
Рекомендации при недостатке данных	Переход на вкладку рекомендаций с менее чем 3 книгами в избранном	Отображение сообщения о необходимости добавить больше книг
Экспорт избранного в PDF	Нажатие кнопки «Экспорт в PDF» при наличии книг	Создание PDF-файла, автоматическое открытие
Экспорт при пустом избранном	Нажатие кнопки «Экспорт в PDF» без книг в избранном	Отображение сообщения «Нет книг для экспорта»
Смена аватара профиля	Выбор корректного изображения, нажатие «Сменить фото»	Обновление аватара, сохранение в базу данных

1.9.2 Результаты тестирования

Сценарий №1 «Регистрация нового пользователя» — при заполнении всех полей регистрационной формы и нажатии кнопки «Завершить регистрацию» система успешно создает новую учетную запись в базе данных. Пользователю отображается информационное сообщение с подтверждением регистрации, после чего он может войти в систему. Для пользователей младше 18 лет фильтр контента принудительно включается и блокируется от изменения, что соответствует требованиям возрастных ограничений. Результат выполнения сценария показан на рисунке 30.

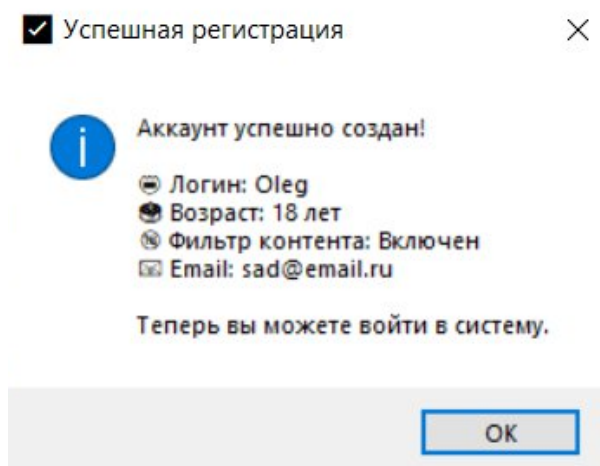


Рисунок 30 — Успешная регистрация нового пользователя

Сценарий №2 «Регистрация с существующим логином» — при попытке зарегистрировать пользователя с логином, который уже присутствует в базе данных, система обнаруживает нарушение ограничения уникальности и не выполняет вставку записи. Пользователю выводится сообщение «Пользователь с таким логином уже существует!», предлагающее выбрать другое имя. Реакция системы на попытку дублирования логина показана на рисунке 31.

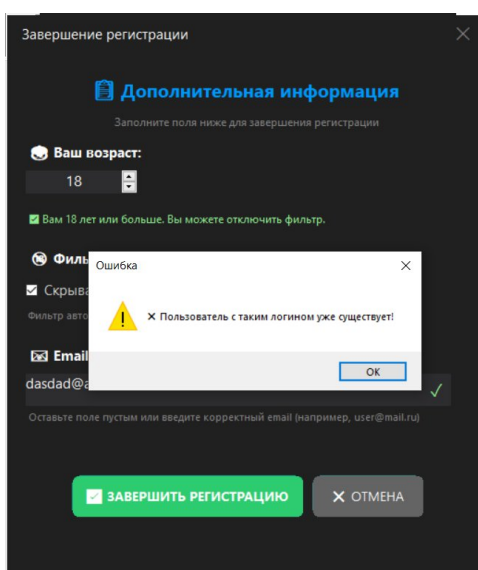


Рисунок 31 — Сообщение об ошибке при регистрации с существующим
ЛОГИНОМ

Сценарий №3 «Аутентификация пользователя» — при вводе корректных учетных данных (логин и пароль) и нажатии кнопки «Войти» система выполняет проверку через метод `ValidateUser`, после чего открывается главное окно приложения с приветствием пользователя. Успешный вход в систему показан на рисунке 32.

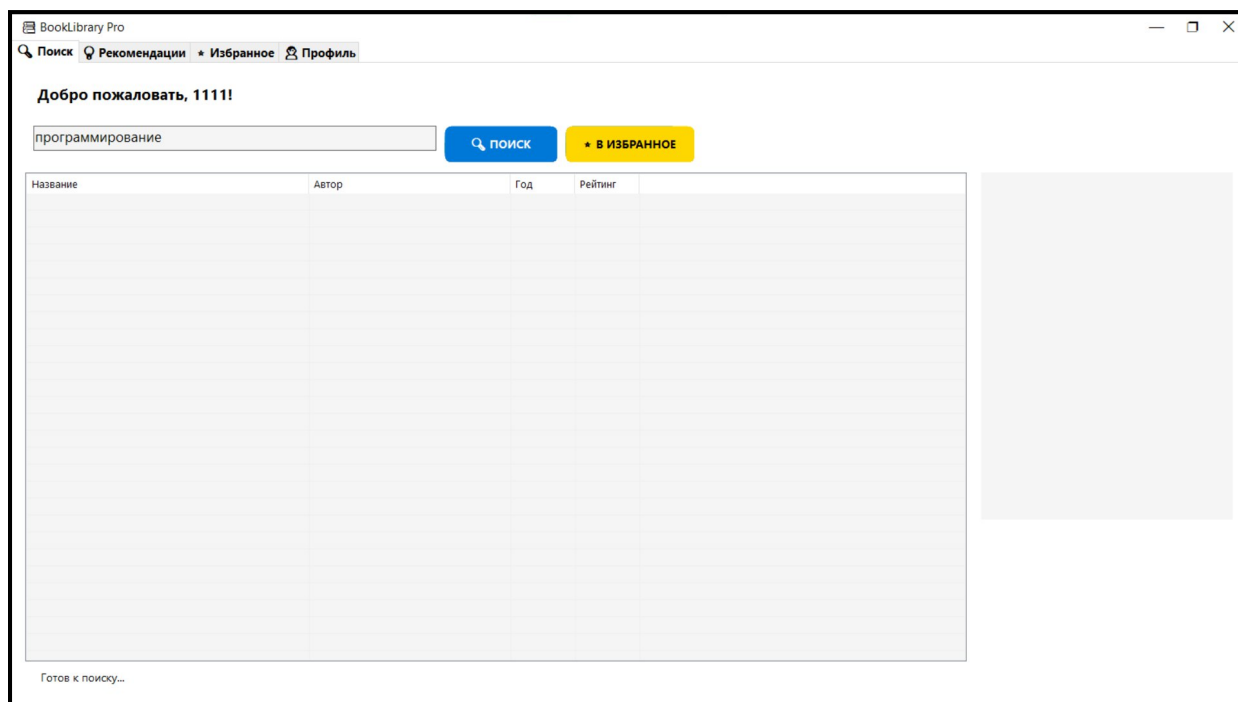


Рисунок 32 — Успешная аутентификация пользователя

Сценарий №4 «Аутентификация с неверными данными» — при вводе неверного логина или пароля метод `ValidateUser` возвращает отрицательный результат, и система отображает диалоговое окно с сообщением «Неверный логин или пароль!». Пользователь остается на форме аутентификации и может повторить попытку ввода. Обработка ошибки аутентификации показана на рисунке 33.

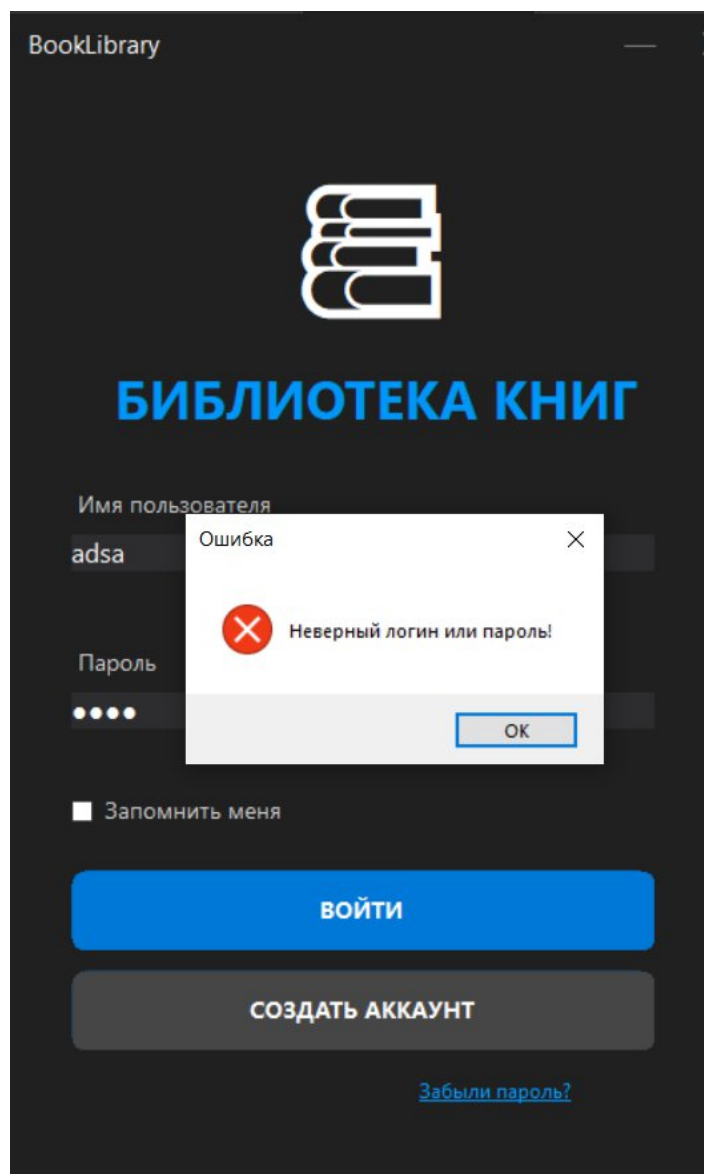


Рисунок 33 — Сообщение об ошибке при неверной аутентификации

Сценарий №5 «Поиск и фильтрация книг» — при вводе поискового запроса в текстовое поле на вкладке «Поиск» система отправляет асинхронный запрос к API OpenLibrary и отображает результаты в таблице. При включенном фильтре запрещенного контента книги с ключевыми запретными словами исключаются из результатов, а в строке статуса отображается количество скрытых книг. Результат поиска с активным фильтром показан на рисунке 34.

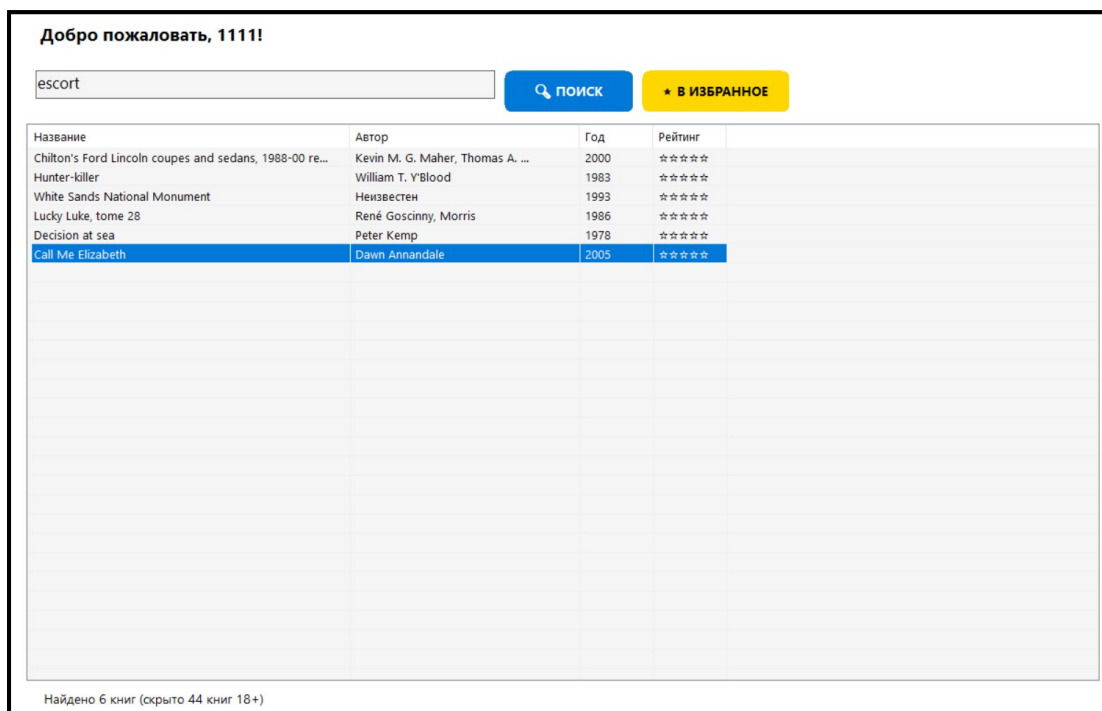


Рисунок 34 — Результат поиска книг с активным фильтром контента

Сценарий №6 «Загрузка некорректного файла аватара» — при попытке загрузить файл неподдерживаемого формата (например, текстовый документ или исполняемый файл) или поврежденное изображение возникает исключение, которое перехватывается блоком обработки ошибок. Система отображает диалоговое окно с описанием причины сбоя, а текущий аватар пользователя остается без изменений. Реакция на ошибочную ситуацию представлена на рисунке 35.

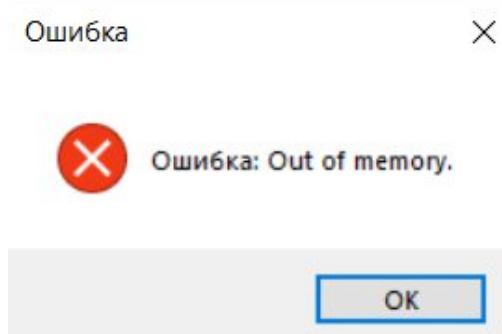


Рисунок 35 — Сообщение об ошибке при загрузке некорректного файла аватара

Сценарий №7 «Добавление книги в избранное» — после выбора книги из результатов поиска и нажатия кнопки «В избранное» система сохраняет запись в таблицу Favorites базы данных. Кнопка меняет цвет и текст на «В избранном», а пользователь получает уведомление об успешном добавлении. Успешное добавление книги показано на рисунке 36.

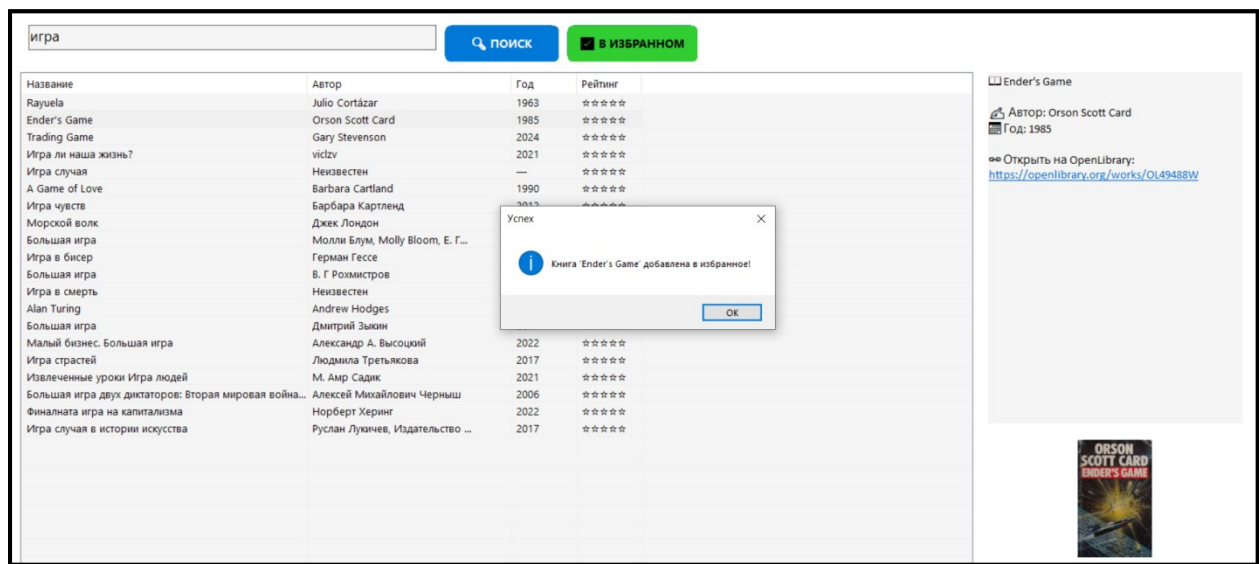


Рисунок 36 — Успешное добавление книги в избранное

Сценарий №8 «Повторное добавление книги» — при попытке добавить книгу, которая уже присутствует в избранном пользователя, система определяет дублирование через проверку ограничения уникальности UNIQUE(Username, BookKey). Вместо повторной вставки пользователю выводится информационное сообщение «Книга уже в избранном!», предотвращающее дублирование данных. Реакция системы показана на рисунке 37.

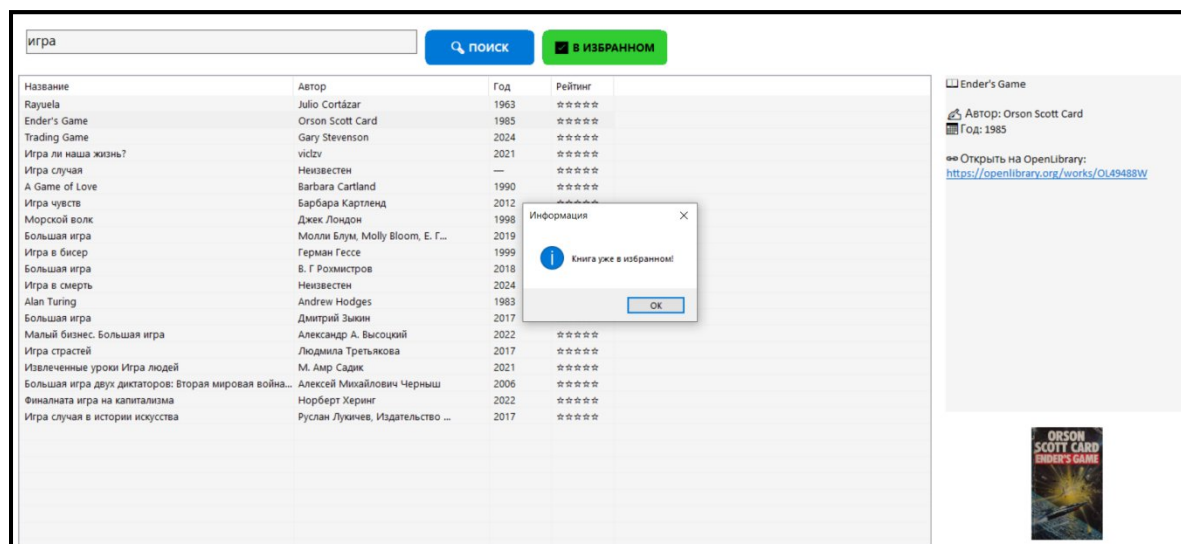


Рисунок 37 — Сообщение при попытке повторного добавления книги

Сценарий №9 «Генерация рекомендаций» — при наличии трех и более книг в избранном и переходе на вкладку «Рекомендации» система выполняет алгоритм коллаборативной фильтрации: находит пользователей со схожими вкусами и формирует подборку книг, отсутствующих в коллекции текущего пользователя. Результаты отображаются в виде карточек с обложками, названиями и кнопками для добавления в избранное. Результат работы рекомендательной системы показан на рисунке 38.

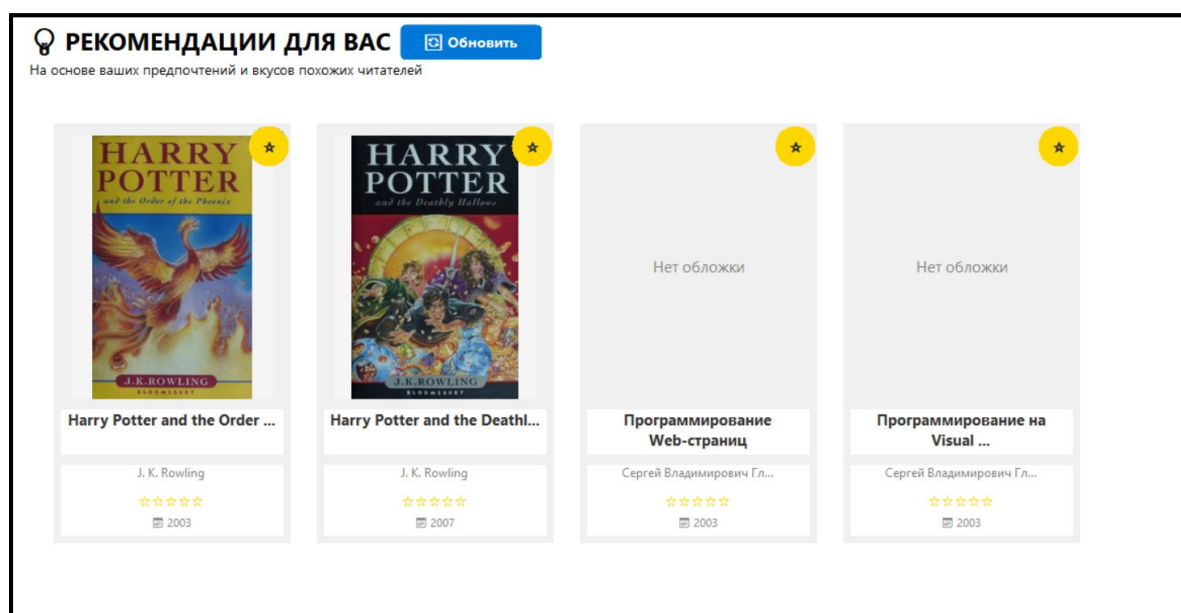


Рисунок 38 — Сформированные персональные рекомендации

Сценарий №10 «Рекомендации при недостатке данных» — если у пользователя в избранном менее трех книг, алгоритм не может достоверно определить предпочтения. Система не выполняет поиск похожих пользователей и отображает информационное сообщение с предложением добавить больше книг для активации рекомендательной функции. Сообщение показано на рисунке 39.

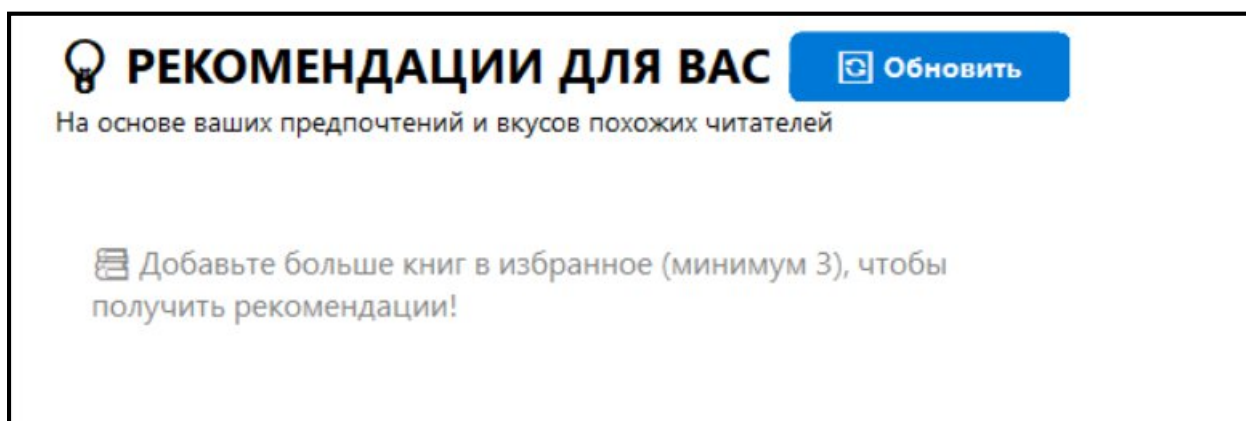


Рисунок 39 — Информационное сообщение при недостаточности данных для рекомендаций

Сценарий №11 «Экспорт избранного в PDF» — при нажатии кнопки «Экспорт в PDF» на вкладке избранного система проверяет наличие книг в коллекции. При положительном результате открывается диалог сохранения файла, после чего формируется структурированный PDF–документ с таблицей избранных книг. По завершении файл автоматически открывается для просмотра. Результат экспорта представлен на рисунке 40.

Список избранных книг пользователя 1111

Название	Автор	Год	Добавлено
Rayuela	Julio Cortázar	1963	19.05.2026 22:14
Большая игра	Молли Блум, Molly Bloom, Е. Г. Богданова	2019	19.05.2026 22:14
Игра в бисер	Герман Гессе	1999	19.05.2026 22:14
Большая игра двух диктаторов: Вторая мировая война - до и после	Алексей Михайлович Черныш	2006	19.05.2026 22:14
Trading Game	Gary Stevenson	2024	19.05.2026 22:14
Игра в смерть	Неизвестен	2024	19.05.2026 22:14
Игра страстей	Людмила Третьякова	2017	19.05.2026 22:14
Игра случая в истории искусства	Руслан Лукичев, Издательство "Алетейя"	2017	19.05.2026 22:14
Большая игра	Дмитрий Зыкин	2017	19.05.2026 22:14
Alan Turing	Andrew Hodges	1983	19.05.2026 22:14

Экспортировано: 19.05.2026 22:14

Рисунок 40 — Сформированный PDF–документ со списком избранных книг

Сценарий №12 «Экспорт при пустом избранном» — если пользователь пытается экспортировать список избранного, не имея ни одной сохраненной книги, система не открывает диалог сохранения и не формирует PDF–файл. Вместо этого отображается предупреждающее сообщение «Нет книг для экспорта!», информирующее пользователя о необходимости сначала добавить книги в избранное. Сообщение показано на рисунке 41.

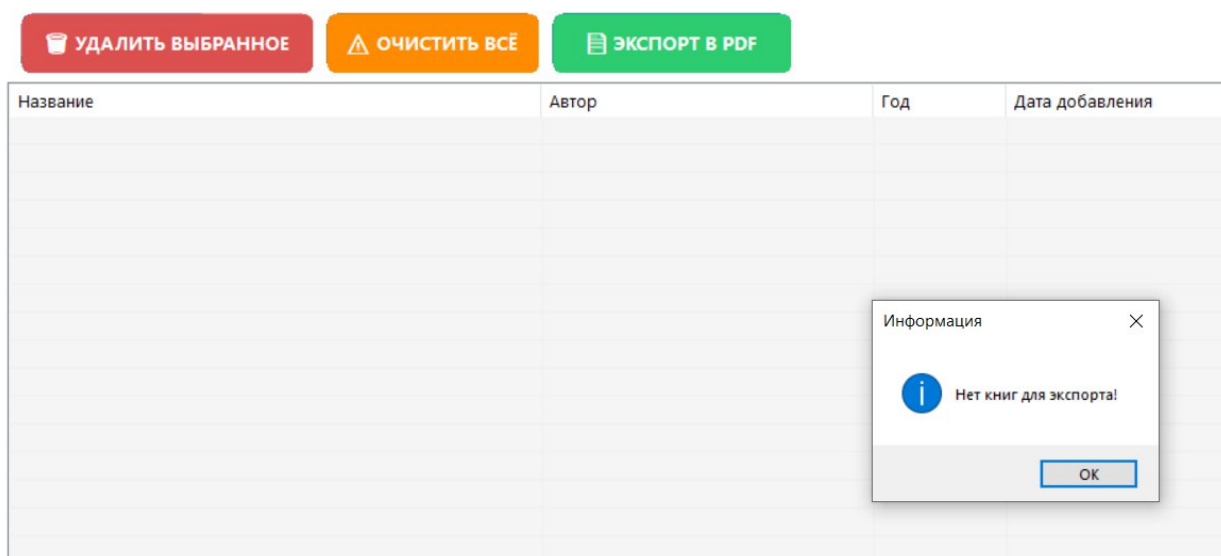


Рисунок 41 — Предупреждение при попытке экспорта пустого избранного

Сценарий №13 «Смена аватара профиля» — при нажатии на «Сменить фото» на вкладке профиля открывается диалог выбора файла с фильтрацией по графическим форматам. После выбора корректного изображения система масштабирует его до требуемого размера, сохраняет оригинал в базу данных и обновляет отображение на форме. Успешное выполнение операции с подтверждающим уведомлением показано на рисунке 42.

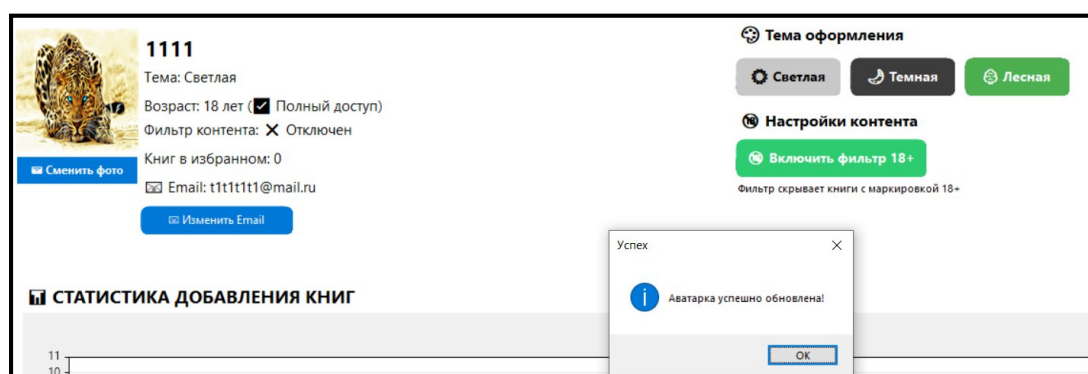


Рисунок 42 — Успешная смена аватара профиля

1.9.3 Вывод по результатам тестирования

В ходе выполнения всех тестовых сценариев разработанное программное обеспечение продемонстрировало корректную работу и соответствие заявленным требованиям. Все проверяемые функции выполняются без ошибок и в полном объеме.

Особого внимания заслуживает корректная обработка неудачных сценариев. Во всех исключительных ситуациях пользователю выводятся понятные информационные сообщения, а приложение продолжает стабильную работу без сбоев и потери данных.

Таким образом, по результатам проведенного тестирования можно сделать вывод о том, что разработанный дипломный проект готов к эксплуатации и полностью соответствует техническому заданию.

2 Требования безопасности по охране труда при работе с ПЭВМ

В этом разделе рассматриваются требования охраны труда и техники безопасности при эксплуатации персональных компьютеров и разработке программного обеспечения, а также способы минимизации негативного влияния этих процессов на здоровье человека.

2.1 Общие требования безопасности

Работать за персональным компьютером рекомендуется только при отсутствии медицинских противопоказаний. Желательно предварительно ознакомиться с правилами безопасного обращения с техникой и основами электробезопасности.

При длительном использовании компьютера на человека могут воздействовать следующие неблагоприятные факторы:

- повышенный уровень электромагнитных излучений;
- накопление статического электричества;
- снижение ионизации воздуха;
- длительные статические нагрузки на тело;
- перенапряжение зрения.

При работе за компьютером или ноутбуком в домашних условиях рекомендуется:

- знать и соблюдать рекомендации производителя по эксплуатации устройства;
- придерживаться разумного режима труда и отдыха, делать перерывы, соблюдать личную гигиену, а также своевременно проходить медицинские осмотры для контроля состояния здоровья (прежде всего зрения и опорно–двигательного аппарата)[4];

– знать, где находится домашняя аптечка. При обнаружении неисправностей компьютера или его периферии (принтера, сканера, клавиатуры, сетевых устройств, блока бесперебойного питания и прочего) прекратить использование до устранения проблем[4];

– при возникновении любой ситуации, угрожающей жизни или здоровью (например, искрение, запах гари), немедленно отключить технику от сети и принять меры предосторожности;

– использовать компьютер в соответствии с его назначением, избегая действий, способных привести к поломке или травме;

– поддерживать порядок на рабочем месте в чистоте и соблюдать элементарные правила пожарной безопасности (не закрывать вентиляцию, не ставить рядом с нагревательными приборами, не перегружать розетки)[4].

Если в комнате находится несколько компьютеров, расстояние от экрана одного монитора до задней панели другого желательно выдерживать не менее 2,0 м, а расстояние между боковыми сторонами мониторов — не менее 1,2 м. Рабочее место с компьютером лучше располагать так, чтобы естественный свет из окна падал сбоку, преимущественно слева. Окна в комнате, где используется компьютер, желательно оборудовать регулируемыми средствами затемнения: жалюзи, плотными шторами или внешними козырьками.

Мебель для работы с компьютером дома должна быть по возможности удобной и отвечать следующим рекомендациям:

– высота рабочей поверхности стола — в пределах 680–800 мм (или около 725 мм при отсутствии регулировки);

– под столом должно быть достаточно пространства для ног: высотой не менее 600 мм, глубиной на уровне колен — не менее 450 мм, для вытянутых ног — не менее 650 мм;

– стул лучше использовать с регулировкой высоты, угла наклона сиденья и спинки, а также положения спинки относительно сиденья;

– желательно иметь подставку для ног шириной от 300 мм и глубиной от 400 мм, с возможностью настройки по высоте до 150 мм и углу наклона до 20°; поверхность подставки должна быть нескользящей, с небольшим бортиком спереди высотой около 10 мм;

– полезно оборудовать рабочее место подставкой для документов (пюпитром), которую легко перемещать;

– не рекомендуется загромождать рабочее пространство лишними вещами, бумагами[4];

– важно оставлять свободными проходы и доступ к средствам пожаротушения.

Искусственное освещение в комнате лучше делать общим и равномерным. Если приходится много работать с бумажными документами, дополнительно полезно использовать местный светильник, направленный на зону документов.

Расстояние от глаз до экрана монитора должно составлять около 600–700 мм (не ближе 500 мм), в зависимости от размера символов и комфортности восприятия. Для поддержания здорового микроклимата помещение, где стоит компьютер, желательно проветривать примерно каждые 2 часа. Беременным и кормящим женщинам рекомендуется по возможности ограничивать время пребывания за компьютером, а при необходимости — консультироваться с врачом.

Пренебрежение данными рекомендациями может приводить к повышенной утомляемости, ухудшению самочувствия и потенциальному вреду для здоровья, поэтому к организации домашнего компьютерного места стоит отнестись ответственно.

2.2 Электробезопасность

Основными нормативными документами, содержащими обязательные требования, являются:

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		61

– Межгосударственный стандарт ГОСТ 12.1.019–2017содержащий перечень основных вредных и опасных производственных факторов и их классификационные характеристики;

– Типовая инструкция по охране труда при работе на персональном компьютере ТОО Р–45–084–01[5].

Первоочередной задачей является приведение параметров помещения в соответствие с установленными нормами. Относительная влажность воздуха не должна превышать 75%, температура — 30 °С. Необходимо исключить наличие токопроводящей пыли и убедиться в электробезопасности напольного покрытия. Компьютер следует размещать таким образом, чтобы в зоне досягаемости пользователя отсутствовали металлические предметы и конструкции (радиаторы отопления, трубопроводы и т.д.).

Перед установкой и началом эксплуатации компьютерной техники надлежит проверить исправность электропроводки, розеток и выключателей. Перегрузка одной розетки путём одновременного подключения всего оборудования не допускается — нагрузку следует распределять равномерно на несколько электрических точек. Рекомендуются заземление корпуса ПК. В процессе эксплуатации требуется осуществлять постоянный контроль целостности изоляции силовых кабелей: не допускаются их перегибы, переламывание и чрезмерное механическое воздействие. Извлечение вилки из розетки производится только путём захвата корпуса вилки, но не за провод.

Работа за компьютером должна осуществляться только сухими руками. Категорически запрещается попадание любой жидкости на корпус системного блока и периферийных устройств. Не допускается частое и необоснованное включение и выключение техники. Очистка от пыли и протирка включённого компьютера или экрана запрещены. Не разрешается размещать на компьютерной технике посторонние предметы и перекрывать вентиляционные отверстия оборудования.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		62

При возникновении малейшего подозрения на неисправность любого компонента ПК эксплуатацию оборудования следует немедленно прекратить и отключить его от питающей сети. Самостоятельный ремонт неисправного компьютера без наличия соответствующего опыта и профессиональных знаний не допускается. Все диагностические и ремонтные работы проводятся исключительно при полностью отключённом питании.

2.3 Пожарная безопасность

Первоочередной задачей является приведение параметров помещения в соответствие с установленными нормами. Относительная влажность воздуха не должна превышать 75%, температура — 30 °С. Необходимо исключить наличие токопроводящей пыли и убедиться в электробезопасности напольного покрытия. Компьютер следует размещать таким образом, чтобы в зоне досягаемости пользователя отсутствовали металлические предметы и конструкции (радиаторы отопления, трубопроводы и т.д.).

Перед установкой и началом эксплуатации компьютерной техники надлежит проверить исправность электропроводки, розеток и выключателей. Перегрузка одной розетки путём одновременного подключения всего оборудования не допускается — нагрузку следует распределять равномерно на несколько электрических точек. Рекомендуются заземление корпуса ПК. В процессе эксплуатации требуется осуществлять постоянный контроль целостности изоляции силовых кабелей: не допускаются их перегибы, переламывание и чрезмерное механическое воздействие. Извлечение вилки из розетки производится только путём захвата корпуса вилки, но не за провод.

Работа за компьютером должна осуществляться только сухими руками. Категорически запрещается попадание любой жидкости на корпус системного блока и периферийных устройств. Не допускается частое и необоснованное включение и выключение техники. Очистка от пыли и протирка включённого компьютера или экрана запрещены. Не разрешается размещать на

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		63

компьютерной технике посторонние предметы и перекрывать вентиляционные отверстия оборудования[6].

При возникновении малейшего подозрения на неисправность любого компонента ПК эксплуатацию оборудования следует немедленно прекратить и отключить его от питающей сети. Самостоятельный ремонт неисправного компьютера без наличия соответствующего опыта и профессиональных знаний не допускается. Все диагностические и ремонтные работы проводятся исключительно при полностью отключённом питании.

При эксплуатации электронно–вычислительных машин не исключена опасность различного рода возгораний. В современных компьютерах отмечается высокая плотность размещения элементов электронных систем; в непосредственной близости друг от друга располагаются соединительные провода и коммуникационные кабели. При протекании электрического тока по ним выделяется значительное количество теплоты, что способно привести к повышению температуры отдельных узлов до 80–100 °С. При этом возможны оплавление изоляции соединительных проводов, их оголение и, как следствие, короткое замыкание, сопровождаемое искрением, которое ведёт к недопустимым перегрузкам элементов электронных схем. Перегреваясь, они сгорают с разбрызгиванием искр. Для отвода избыточного тепла от ЭВМ предусмотрены системы вентиляции и кондиционирования воздуха. Вместе с тем данные системы представляют дополнительную пожарную опасность, поскольку, с одной стороны, воздуховоды обеспечивают подачу кислорода, являющегося окислителем, во все помещения, а с другой — при возникновении пожара способствуют быстрому распространению огня и продуктов горения по всем помещениям и устройствам, с которыми они связаны.

Питание к электроустановкам подаётся по кабельным линиям, представляющим особую пожарную опасность. Наличие горючего изоляционного материала, вероятных источников зажигания в виде

электрических искр и дуг, разветвлённость и труднодоступность делают кабельные линии местами наиболее вероятного возникновения и развития пожара. Эксплуатация ЭВМ сопряжена с необходимостью проведения обслуживающих, ремонтных и профилактических работ, в ходе которых применяются различные смазочные материалы, легковоспламеняющиеся жидкости, прокладываются временные электропроводки, ведутся пайка и чистка отдельных узлов. Возникающая при этом дополнительная пожарная опасность требует реализации соответствующих мер пожарной профилактики. Для предупреждения возгорания все виды кабелей надлежит прокладывать в металлических газонаполненных трубах. Ручные углекислотные огнетушители размещаются из расчёта один огнетушитель на 40–50 м².

В случае пожара срабатывает находящаяся в помещениях автоматическая установка пожаротушения, наиболее распространёнными из которых являются газовые автоматические установки пожаротушения, оснащённые световой и звуковой сигнализацией.

Для предотвращения распространения огня из одной части здания в другую устраиваются противопожарные преграды в виде противопожарных стен, перегородок, перекрытий, зон, тамбуров–шлюзов, дверей, окон, люков и клапанов.

В здании на случай возникновения пожара предусматривается не менее двух эвакуационных выходов; при этом через машинный зал, также имеющий не менее двух выходов, не должны проходить пути эвакуации сотрудников иных подразделений. В прочих производственных помещениях допускается проектировать один выход при условии, что расстояние от наиболее удалённого рабочего места до выхода не превышает 25 м, а количество одновременно работающих в смене — не более 25 человек. Проходы, коридоры и рабочие места не должны загромождаться архивными материалами и бумагой. На эвакуационных путях предусматривается как естественное, так и искусственное аварийное освещение.

					ПГУ1.090301.11.001 ПЗ	Лист
						65
Изм.	Лист	№ докум.	Подпись	Дата		

Для хранения носителей информации применяются негорючие металлические шкафы; двери в хранилище также выполняются в негорючем исполнении. Комплекс организационных и технических мероприятий пожарной профилактики позволяет предотвратить пожар, а в случае его возникновения — обеспечить безопасность людей, ограничить распространение огня и создать условия для успешного тушения.

В контексте обеспечения безопасности пользователя следует упомянуть европейские стандарты ТСО95 и ТСО99 — рекомендации, разработанные Шведской конференцией профсоюзов и Национальным советом индустриального и технического развития Швеции, регламентирующие взаимодействие оборудования с окружающей средой. Данные стандарты требуют снижения электрических и магнитных полей до технически достижимого уровня с целью защиты пользователя. Для получения сертификата ТСО95 электротехническое изделие должно соответствовать критериям низкого излучения, то есть обладать пониженным уровнем электромагнитного поля, обеспечивать автоматическое снижение энергопотребления при длительном неиспользовании, а также отвечать европейским стандартам пожарной и электрической безопасности.

При возникновении пожара обязательному срабатыванию подлежит система оповещения и управления эвакуацией людей (СОУЭ). Оповещение и управление эвакуацией людей при пожаре должно осуществляться одним из следующих способов или их комбинацией:

- подачей звуковых и (или) световых сигналов во все помещения здания с постоянным или временным пребыванием людей;(сдлать норм переносы)
- трансляцией текстов о необходимости эвакуации, путях эвакуации, направлении движения и других действиях, направленных на обеспечение безопасности людей;
- трансляцией специально разработанных текстов, направленных на предотвращение паники и иных явлений, усложняющих эвакуацию;

- размещением эвакуационных знаков безопасности на путях эвакуации;
- включением эвакуационных знаков безопасности;
- включением эвакуационного освещения;
- дистанционным открыванием дверей эвакуационных выходов (например, оборудованных электромагнитными замками);
- связью пожарного поста–диспетчерской с зонами пожарного оповещения.

СОУЭ проектируется с целью реализации планов эвакуации. При проектировании должна предусматриваться возможность сопряжения СОУЭ с системой оповещения гражданской обороны. Включение СОУЭ осуществляется от командного импульса, формируемого автоматической установкой пожарной сигнализации или пожаротушения. Допускается применение дистанционного и местного включения, если в соответствии с нормативными документами для данного вида зданий не требуется оснащение автоматическими установками пожаротушения и автоматической пожарной сигнализацией. Пусковые элементы должны быть выполнены и размещены в соответствии с требованиями, предъявляемыми к ручным пожарным извещателям.

2.4 Режим труда и отдыха

В СанПиН есть понятие суммарного времени регламентированных перерывов, которое зависит от категории трудовой деятельности и уровня нагрузки за рабочую смену при работе с компьютером (п. 1.2 Приложения № 7 к СанПиН 2.2.2/2.4.1340–03). При 8–часовой рабочей смене суммарное время перерывов составляет от 50 до 90 минут. 12–часовой рабочий день вынуждает установить перерывы суммарной продолжительностью от 80 до 140 минут. К примеру, если человек в течение 8–часового рабочего дня работает за компьютером 50% рабочего времени (то есть до 4 часов), то суммарные

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		67

перерывы для отдыха от ПЭВМ должны составлять 70 минут[7]. При работе за компьютером ночью (с 22 до 6 часов) продолжительность регламентированных перерывов следует увеличить на 30% (п. 1.6 Приложения № 7 к СанПиН 2.2.2/2.4.1340–03). Указанные перерывы включаются в рабочее время. То есть они не продлевают продолжительность рабочего дня сотрудника. Во время этих перерывов работник не должен выполнять другую работу. Перерыв предоставляется ему для отдыха (Письмо Минтруда от 14.06.2017 № 14–2/ООГ–4765). Кроме того, важно помнить, что перерывы в работе для отдыха от компьютера предоставляются отдельно от перерыва на обед (ст. 108, 109 ТК РФ).

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		68

Заключение

В процессе разработки информационной системы поиска и каталогизации авторов на основе анализа названий книг были получены практические навыки создания десктопных Windows–приложений с использованием Windows Forms и языка C#. В ходе работы освоены технологии работы с базами данных SQLite для хранения пользовательских данных, избранных книг и кэширования обложек. Изучены принципы интеграции с внешними API на примере Open Library, включая обработку JSON–ответов, асинхронные HTTP–запросы и кэширование изображений.

За период выполнения работы реализованы: система регистрации и аутентификации пользователей с возможностью восстановления пароля через email, персонализированные профили с настраиваемыми темами оформления и возрастными ограничениями, поисковая система с фильтрацией запрещенного контента, по ключевым словам, система избранного с экспортом в PDF и визуальной статистикой чтения, а также рекомендательная система на основе коллаборативной фильтрации предпочтений пользователей. Разработан современный пользовательский интерфейс с кастомными элементами управления, поддержкой нескольких цветовых схем и адаптивной версткой.

Приложение обеспечивает удобный доступ к каталогу Open Library с возможностью быстрого поиска, просмотра детальной информации о книгах и добавления их в персональную библиотеку. В дальнейшем проект может быть расширен путем добавления интеграции с другими книжными API, внедрения социальных функций (обзоры, рейтинги, рекомендации друзьям), реализации облачной синхронизации данных между устройствами, а также разработки мобильной версии приложения.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		69

Список использованных источников

1. Яндекс–Книги Википедия [Электронный ресурс] /Режимдоступа: https://ru.wikipedia.org/wiki/%D0%AF%D0%BD%D0%B4%D0%B5%D0%BA%D1%81_%D0%9A%D0%BD%D0%B8%D0%B3%D0%B8

(датаобращения01.05.2026)

2. Троелсен Э., Джепикс Ф. Язык программирования C# 7 и платформы. NET и. NET Core. – Litres, 2019. [Электронный ресурс]

Режим доступа: <https://books.google.com/books?hl=ru&lr=&id=wDrBDwAAQBAJ&oi=fnd&pg=PA32&dq=%D1%8F%D0%B7%D1%8B%D0%BA+c%D1%81%D0%B8%D0%B3%D0%B8&ots=ogRllzgfvd&sig=emom1JCEcb6UsZaiBTZlQo96h8o>свободный

(датаобращения: 01.05.2026).

3. Пьянова Е. А. и др. Обзор новых возможностей Visual Studio 2022 //Теплотехника и информатика в образовании, науке и производстве (ТИМ'2022).—Екатеринбург, 2022. – 2022. – С. 280-285 [Электронный ресурс]

/ Режим

доступа:<https://elar.urfu.ru/handle/10995/119249>свободный(датаобращения:01.05.2026).

4. Гузенко Е. Ю., Мартынов И. С., Мисюряев В. Ю. ПОВЫШЕНИЕ БЕЗОПАСНОСТИ ТРУДА ОПЕРАТОРОВ ПК //Новая наука: Современное состояние и пути развития. – 2015. – №. 3. – С. 80-83. [Электронныйресурс].URL:<https://elibrary.ru/item.asp?id=24235002>

(датаобращения01.05.2026)

5. Гралько В. В. Электробезопасность компьютеров и компьютерных сетей. – 2010.

[Электронныйресурс].URL:<https://rep.bntu.by/bitstream/handle/data/111392/102-105.pdf?sequence=1> (датаобращения01.05.2026)

6. ЛОНЧАКОВ И. А., СУХАЕВА А. Р. БЕЗОПАСНОСТЬ В ТЕХНОСФЕРЕ. – Иркутский государственный аграрный университет им. АА

Ежевского КОНФЕРЕНЦИЯ: БЕЗОПАСНОСТЬ В ТЕХНОСФЕРЕ Иркутск, 27
марта 2026 года. [Электронный
ресурс]Режимдоступа:<https://elibrary.ru/item.asp?id=82450107>свобо
дный.(датаобращения:01.05.2026).

7. Гигиенические требования к персональным
электронно–вычислительным машинам и организации работы. Изменение N 3
к СанПиН 2.2.2/2.4.1340-03[Электронный
ресурс]Режимдоступа:https://ohranatruda.ru/ot_biblio/norma/249905/
свободный.(датаобращения:01.05.2026).

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		71

Приложение А

(обязательное)

«Информационная система управления персональными подборками
литературных произведений»

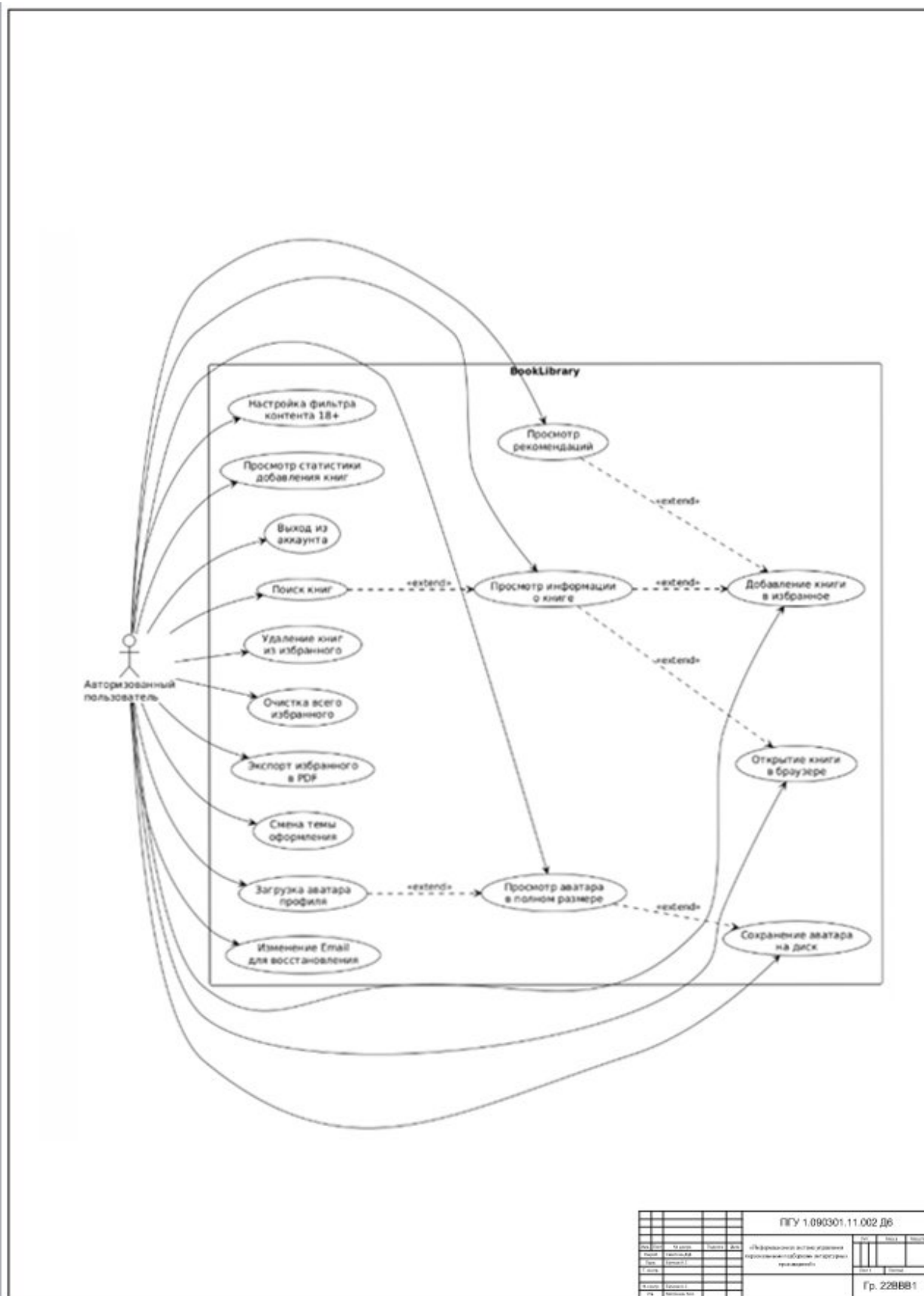
UML диаграммы и схемы

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		72

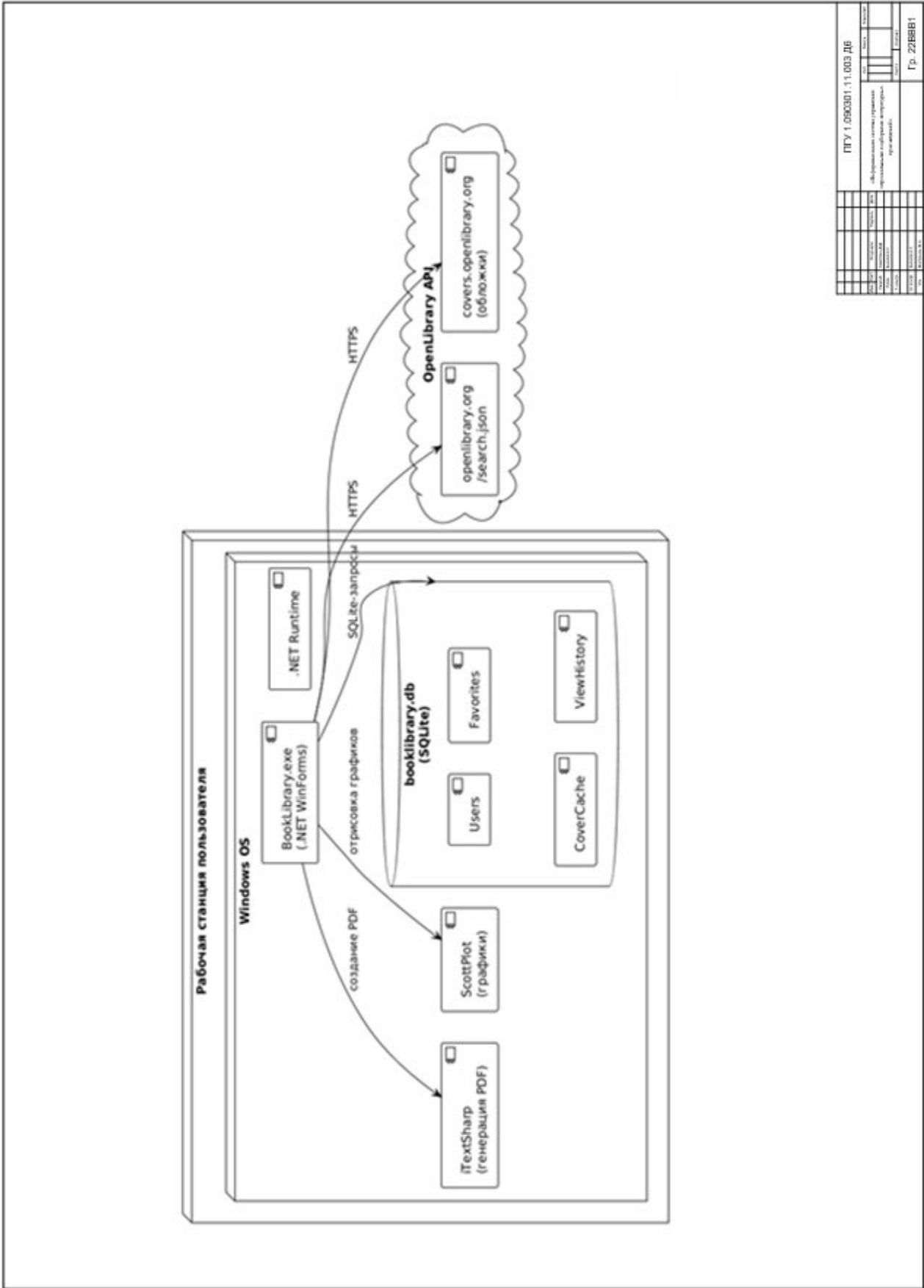
ПОЛЬЗОВАТЕЛЯ



Приложение А.2 – Диаграмма прецедентов авторизованного пользователя



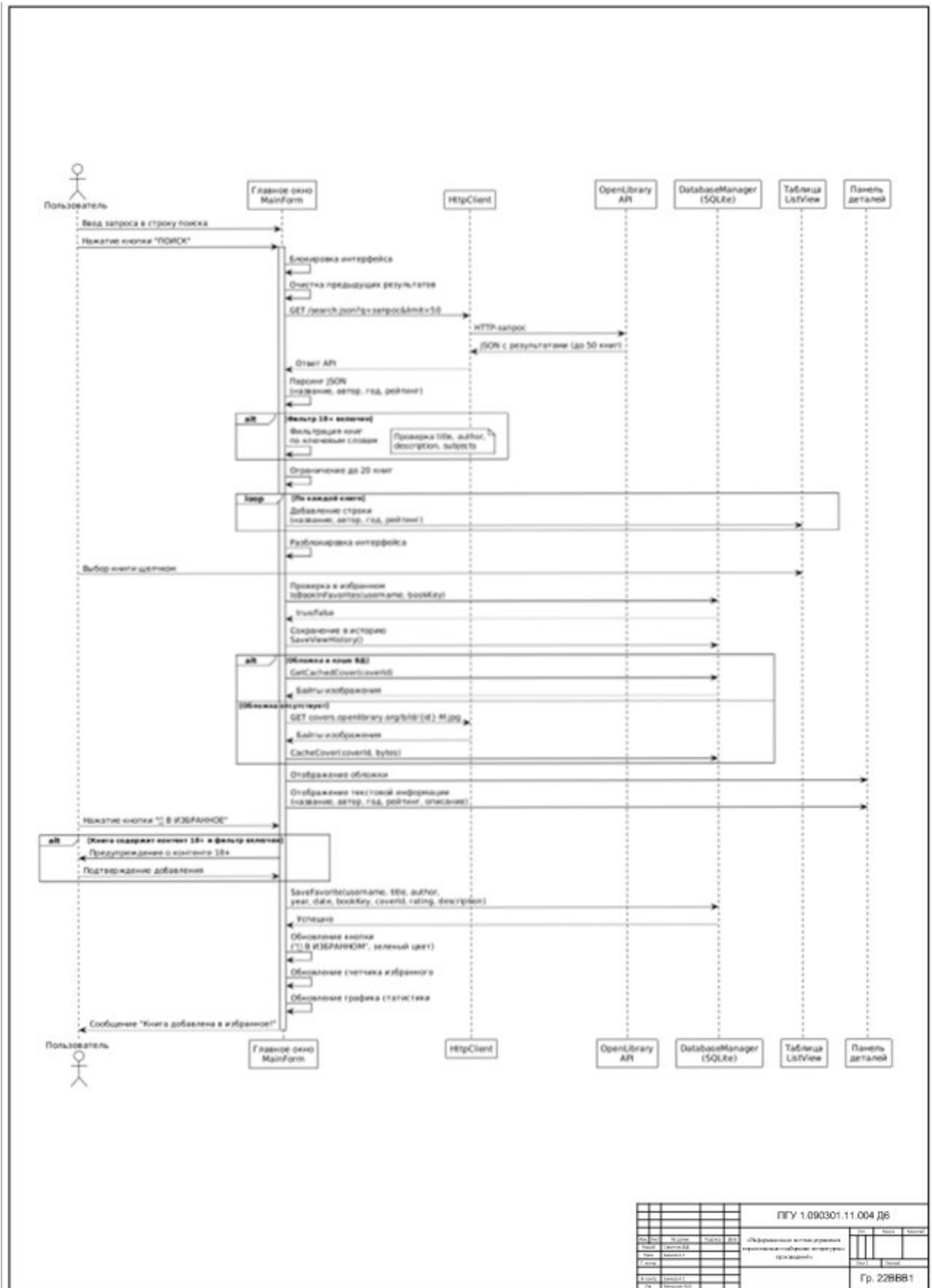
Приложение А.3 – Диаграмма развёртывания



ПГУ 1.090301.11.003 Д6	
Исполнитель	Исполнитель
Проверка	Проверка
Дата	Дата
Гр. 2288В1	

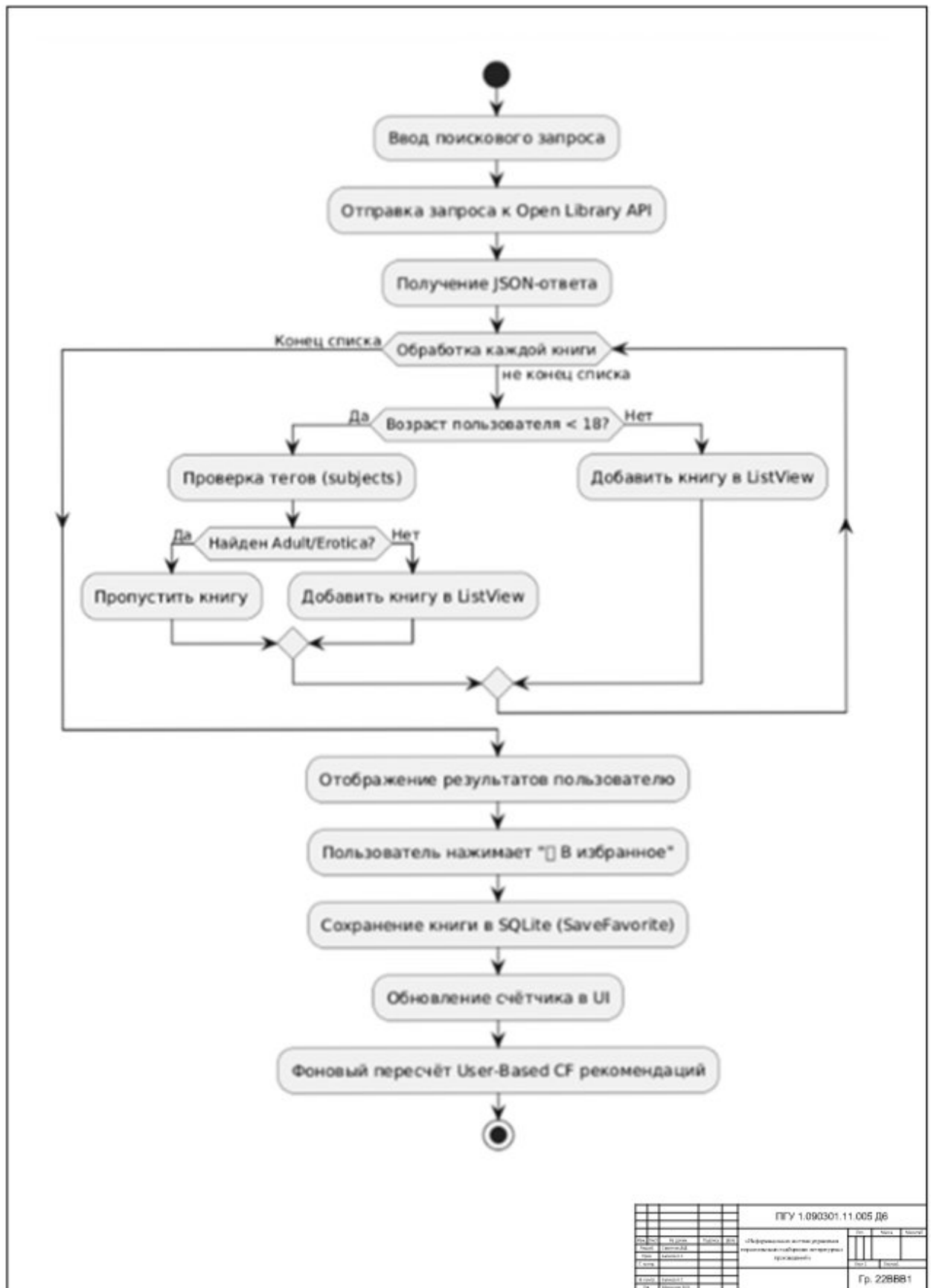
Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.4—Диаграмма последовательностей



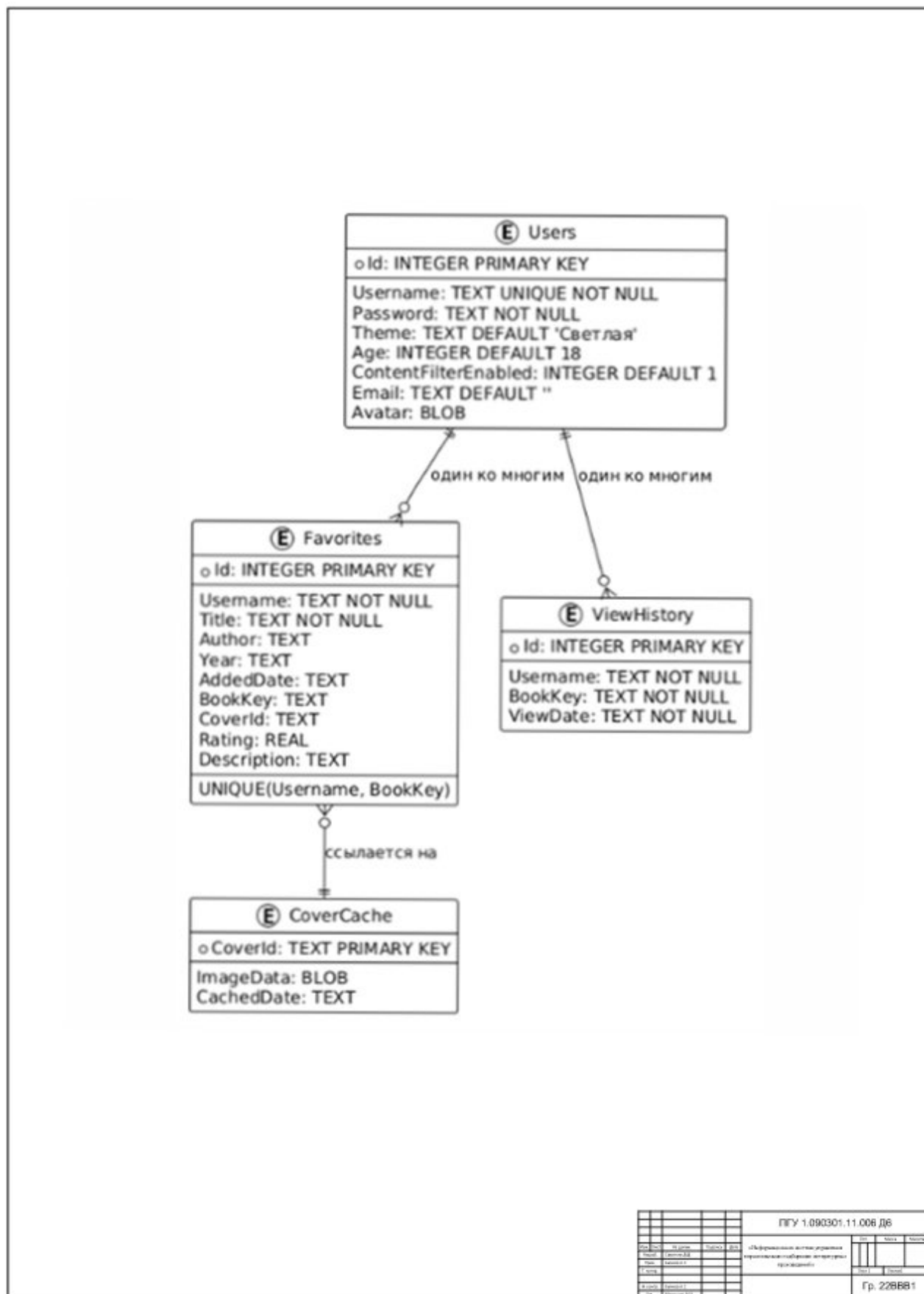
ПГУ 1.090301.11.004 Д6									
Имя	Место	Дата	Время	Страна	Город	Улица	Дом	Почта	Телефон
Имя	Место	Дата	Время	Страна	Город	Улица	Дом	Почта	Телефон
Имя	Место	Дата	Время	Страна	Город	Улица	Дом	Почта	Телефон
Имя	Место	Дата	Время	Страна	Город	Улица	Дом	Почта	Телефон
Имя	Место	Дата	Время	Страна	Город	Улица	Дом	Почта	Телефон
Имя	Место	Дата	Время	Страна	Город	Улица	Дом	Почта	Телефон
Имя	Место	Дата	Время	Страна	Город	Улица	Дом	Почта	Телефон
Имя	Место	Дата	Время	Страна	Город	Улица	Дом	Почта	Телефон
Имя	Место	Дата	Время	Страна	Город	Улица	Дом	Почта	Телефон

Приложение А.5–Функциональная схема работы проекта



ПГУ 1.090301.11.005 Д6			
Имя	Фамилия	Имя	Фамилия
Группа	Специальность	Группа	Специальность
Дата	Подпись	Дата	Подпись
Гр. 22ВВВ1			

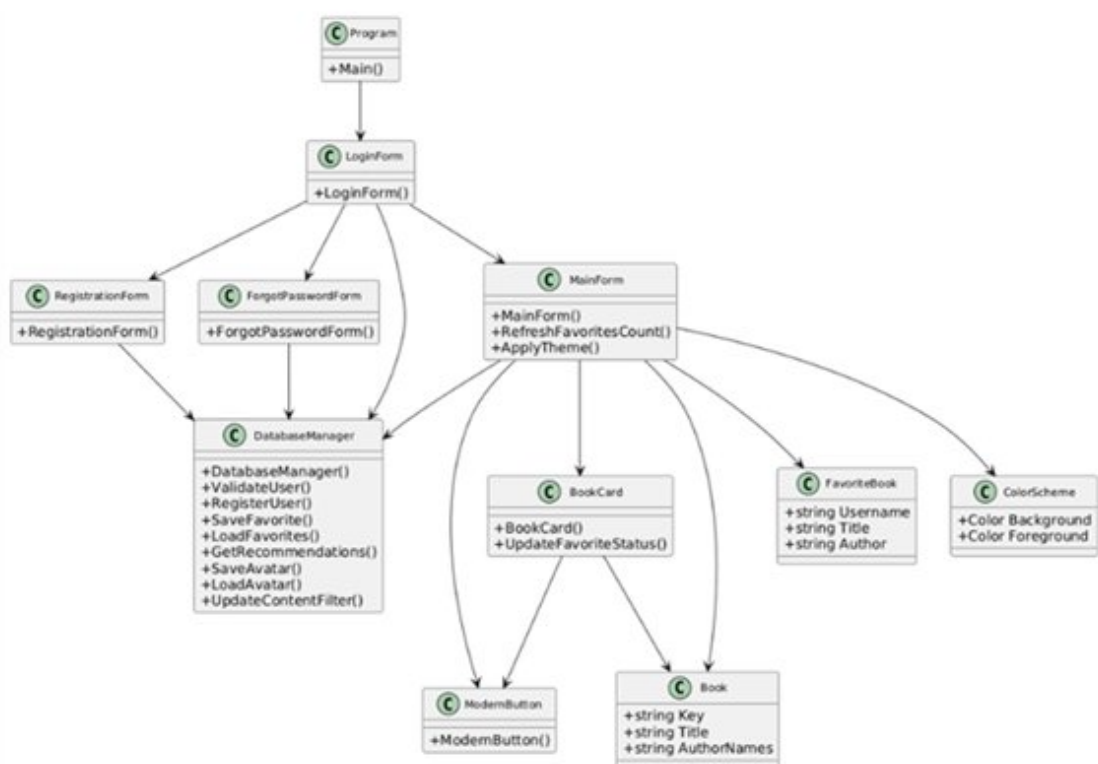
Приложение А.6 – Диаграмма базы данных



ПГУ 1.090301.11.008 ДБ									
Информационная система управления	Информационная система управления	Информационная система управления	Информационная система управления	Информационная система управления	Информационная система управления	Информационная система управления	Информационная система управления	Информационная система управления	Информационная система управления
Гр. 228ВВ1									

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.7 – Диаграмма классов

[illegible]

Приложение Б

(обязательно)

«Информационная система управления персональными подборками
литературных произведений»

Листинг программы

					ПГУ1.090301.11.001 ПЗ	Лист
						80
Изм.	Лист	№ докум.	Подпись	Дата		

```

using System;
using System.Collections.Generic;
using System.Data.SQLite;
using System.Diagnostics;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Runtime.InteropServices;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;
using System.Windows.Forms;
using iTextSharp.text;
using iTextSharp.text.pdf;
using ScottPlot;
using ScottPlot.WinForms;
using Microsoft.VisualBasic;

using WinFormsLabel = System.Windows.Forms.Label;
using WinFormsButton = System.Windows.Forms.Button;
using WinFormsTextBox = System.Windows.Forms.TextBox;
using WinFormsPanel = System.Windows.Forms.Panel;
using WinFormsPictureBox = System.Windows.Forms.PictureBox;
using WinFormsProgressBar = System.Windows.Forms.ProgressBar;
using DrawingFont = System.Drawing.Font;
using DrawingColor = System.Drawing.Color;
using DrawingImage = System.Drawing.Image;
using DrawingRectangle = System.Drawing.Rectangle;

namespace BookLibraryWithAuth
{
    public class BookCard : WinFormsPanel
    {
        private WinFormsPictureBox coverBox;
        private WinFormsLabel titleLabel;
        private WinFormsLabel authorLabel;
        private WinFormsLabel ratingLabel;
        private ModernButton addButton;
        private MainForm.Book book;
        private bool isInFavorites = false;
        private bool isHovered = false;

        public event EventHandler<MainForm.Book>
AddToFavoritesClicked;
        public event EventHandler<MainForm.Book> BookCardClicked;

        public BookCard(MainForm.Book book, DrawingImage cover =
null, bool alreadyInFavorites = false)
        {
            this.book = book;

```

```

        this.isInFavorites = alreadyInFavorites;

        Size = new Size(240, 420);
        BackColor = DrawingColor.White;
        BorderStyle = BorderStyle.None;
        Margin = new Padding(10);
        Cursor = Cursors.Hand;

        MouseEnter += (s, e) => { isHovered = true;
Invalidate(); };
        MouseLeave += (s, e) => { isHovered = false;
Invalidate(); };
        Click += (s, e) => {
            if (e is MouseEventArgs mouseArgs)
            {
                Point clientPoint = mouseArgs.Location;
                if (!addButton.Bounds.Contains(clientPoint))
                    BookCardClicked?.Invoke(this, book);
            }
        };

        coverBox = new WinFormsPictureBox
        {
            Size = new Size(200, 260),
            Location = new Point(20, 15),
            SizeMode = PictureBoxSizeMode.Zoom,
            BackColor = DrawingColor.FromArgb(245, 245, 245),
            Cursor = Cursors.Hand
        };
        coverBox.Click += (s, e) =>
BookCardClicked?.Invoke(this, book);
        coverBox.Image = cover ?? CreatePlaceholderImage(200,
260, "Нетобложки");

        addButton = new ModernButton
        {
            Text = isInFavorites ? "✓" : "★",
            Size = new Size(40, 40),
            Location = new Point(coverBox.Right - 25,
coverBox.Top - 10),
            NormalColor = isInFavorites ?
DrawingColor.FromArgb(76, 175, 80) : DrawingColor.FromArgb(255,
215, 0),
            HoverColor = isInFavorites ?
DrawingColor.FromArgb(100, 200, 100) : DrawingColor.FromArgb(255,
235, 100),
            PressColor = isInFavorites ?
DrawingColor.FromArgb(60, 150, 60) : DrawingColor.FromArgb(230,
190, 0),
            Font = new DrawingFont("Segoe UI", 16,
System.Drawing.FontStyle.Bold),

```

```

        BorderRadius = 20,
        Cursor = Cursors.Hand,
        FlatStyle = FlatStyle.Flat,
        TextAlign = ContentAlignment.MiddleCenter,
        UseVisualStyleBackColor = false,
        ForeColor = isInFavorites ? DrawingColor.White :
DrawingColor.FromArgb(50, 50, 50)
    };
    addButton.Click += (s, e) => { if (!isInFavorites)
AddToFavoritesClicked?.Invoke(this, book); };

    titleLabel = new WinFormsLabel
    {
        Text = Truncate(book.Title, 30),
        Location = new Point(10, 285),
        Size = new Size(220, 45),
        Font = new DrawingFont("Segoe UI", 11,
System.Drawing.FontStyle.Bold),
        TextAlign = ContentAlignment.TopCenter,
        ForeColor = DrawingColor.FromArgb(50, 50, 50),
        Cursor = Cursors.Hand
    };
    titleLabel.Click += (s, e) =>
BookCardClicked?.Invoke(this, book);
    titleLabel.MouseEnter += (s, e) =>
{ titleLabel.ForeColor = DrawingColor.FromArgb(0, 120, 215);
titleLabel.Font = new DrawingFont(titleLabel.Font,
System.Drawing.FontStyle.Bold |
System.Drawing.FontStyle.Underline); };
    titleLabel.MouseLeave += (s, e) =>
{ titleLabel.ForeColor = DrawingColor.FromArgb(50, 50, 50);
titleLabel.Font = new DrawingFont(titleLabel.Font,
System.Drawing.FontStyle.Bold); };

    authorLabel = new WinFormsLabel
    {
        Text = Truncate(book.AuthorNames ?? "Неизвестен",
25),
        Location = new Point(10, 340),
        Size = new Size(220, 25),
        Font = new DrawingFont("Segoe UI", 9),
        ForeColor = DrawingColor.Gray,
        TextAlign = ContentAlignment.TopCenter,
        Cursor = Cursors.Hand
    };
    authorLabel.Click += (s, e) =>
BookCardClicked?.Invoke(this, book);

    ratingLabel = new WinFormsLabel
    {

```

```

        Text = book.Rating.HasValue ?
GetStarRating(book.Rating.Value) : "☆☆☆☆☆",
        Location = new Point(10, 365),
        Size = new Size(220, 25),
        Font = new DrawingFont("Segoe UI", 11),
        ForeColor = DrawingColor.Gold,
        TextAlign = ContentAlignment.TopCenter,
        Cursor = Cursors.Hand
    };
    ratingLabel.Click += (s, e) =>
BookCardClicked?.Invoke(this, book);

    if (book.FirstPublishYear.HasValue)
    {
        var yearLabel = new WinFormsLabel
        {
            Text = $"📅 {book.FirstPublishYear}",
            Location = new Point(10, 390),
            Size = new Size(220, 20),
            Font = new DrawingFont("Segoe UI", 8),
            ForeColor = DrawingColor.Gray,
            TextAlign = ContentAlignment.TopCenter,
            Cursor = Cursors.Hand
        };
        yearLabel.Click += (s, e) =>
BookCardClicked?.Invoke(this, book);
        Controls.Add(yearLabel);
    }

    Controls.Add(coverBox);
    Controls.Add(addButton);
    Controls.Add(titleLabel);
    Controls.Add(authorLabel);
    Controls.Add(ratingLabel);
    addButton.BringToFront();
}

protected override void OnPaint(PaintEventArgs e)
{
    base.OnPaint(e);
    if (isHovered)
        using (var pen = new Pen(DrawingColor.FromArgb(0,
120, 215), 2))
            e.Graphics.DrawRectangle(pen, 1, 1, Width -
3, Height - 3);
        using (var shadowBrush = new
SolidBrush(DrawingColor.FromArgb(15, 0, 0, 0)))
            e.Graphics.FillRectangle(shadowBrush, new
DrawingRectangle(3, 3, Width - 6, Height - 6));
}

```

```

        private DrawingImage CreatePlaceholderImage(int width,
int height, string text)
        {
            Bitmap bmp = new Bitmap(width, height);
            using (Graphics g = Graphics.FromImage(bmp))
            {
                g.Clear(DrawingColor.FromArgb(240, 240, 240));
                using (var font = new DrawingFont("Segoe UI",
11))
                using (var brush = new
SolidBrush(DrawingColor.Gray))
                {
                    var sf = new StringFormat { Alignment =
StringAlignment.Center, LineAlignment = StringAlignment.Center };
                    g.DrawString(text, font, brush, new
DrawingRectangle(0, 0, width, height), sf);
                }
                using (var bookIconFont = new DrawingFont("Segoe
UI", 48))
                using (var bookBrush = new
SolidBrush(DrawingColor.FromArgb(180, 180, 180)))
                {
                    var sf = new StringFormat { Alignment =
StringAlignment.Center, LineAlignment = StringAlignment.Center };
                    g.DrawString(" ", bookIconFont, bookBrush,
new DrawingRectangle(0, -20, width, height), sf);
                }
            }
            return bmp;
        }

        private string GetStarRating(double rating)
        {
            int fullStars = (int)Math.Floor(rating);
            bool halfStar = (rating - fullStars) >= 0.5;
            var stars = new StringBuilder();
            for (int i = 0; i < fullStars; i++)
stars.Append("★");
            if (halfStar) stars.Append("½");
            int emptyStars = 5 - fullStars - (halfStar ? 1 : 0);
            for (int i = 0; i < emptyStars; i++)
stars.Append("☆");
            return stars.ToString();
        }

        private string Truncate(string text, int length) =>
string.IsNullOrEmpty(text) ? "" : (text.Length <= length ? text :
text.Substring(0, length - 3) + "...");

        public void UpdateFavoriteStatus(bool isFavorite)
        {

```

```

        isInFavorites = isFavorite;
        addButton.Text = isFavorite ? "✓" : "★";
        addButton.NormalColor = isFavorite ?
DrawingColor.FromArgb(76, 175, 80) : DrawingColor.FromArgb(255,
215, 0);
        addButton.HoverColor = isFavorite ?
DrawingColor.FromArgb(100, 200, 100) : DrawingColor.FromArgb(255,
235, 100);
        addButton.PressColor = isFavorite ?
DrawingColor.FromArgb(60, 150, 60) : DrawingColor.FromArgb(230,
190, 0);
        addButton.ForeColor = isFavorite ?
DrawingColor.White : DrawingColor.FromArgb(50, 50, 50);
    }
}

public class ModernButton : WinFormsButton
{
    private DrawingColor _normalColor =
DrawingColor.FromArgb(0, 120, 215);
    private DrawingColor _hoverColor =
DrawingColor.FromArgb(0, 150, 250);
    private DrawingColor _pressColor =
DrawingColor.FromArgb(0, 90, 180);
    private DrawingColor _borderColor =
DrawingColor.Transparent;
    private int _borderRadius = 8;
    private bool _isHovered = false;
    private bool _isPressed = false;

    public DrawingColor NormalColor { get => _normalColor;
set { _normalColor = value; BackColor = value; Invalidate(); } }
    public DrawingColor HoverColor { get => _hoverColor; set
=> _hoverColor = value; }
    public DrawingColor PressColor { get => _pressColor; set
=> _pressColor = value; }
    public DrawingColor BorderColor { get => _borderColor;
set => _borderColor = value; }
    public int BorderRadius { get => _borderRadius; set =>
_borderRadius = value; }

    public ModernButton()
    {
        FlatStyle = FlatStyle.Flat;
        FlatAppearance.BorderSize = 0;
        BackColor = _normalColor;
        ForeColor = DrawingColor.White;
        Font = new DrawingFont("Segoe UI", 10,
System.Drawing.FontStyle.Bold);
        Cursor = Cursors.Hand;
        TextAlign = ContentAlignment.MiddleCenter;
    }
}

```

```

        UseVisualStyleBackColor = false;
        SetStyle(ControlStyles.Selectable, false);

        MouseEnter += (s, e) => { _isHovered = true;
BackColor = _hoverColor; Invalidate(); };
        MouseLeave += (s, e) => { _isHovered = false;
_isPressed = false; BackColor = _normalColor; Invalidate(); };
        MouseDown += (s, e) => { _isPressed = true; BackColor
= _pressColor; Invalidate(); };
        MouseUp += (s, e) => { _isPressed = false; BackColor
= _isHovered ? _hoverColor : _normalColor; Invalidate(); };
    }

    protected override void OnPaint(PaintEventArgs pevent)
    {
        pevent.Graphics.SmoothingMode =
SmoothingMode.AntiAlias;
        DrawingColor currentBackColor = _isPressed ?
_pressColor : (_isHovered ? _hoverColor : _normalColor);
        using (var path =
GetRoundedRectangle(ClientRectangle, _borderRadius))
            using (var brush = new SolidBrush(currentBackColor))
                pevent.Graphics.FillPath(brush, path);
        if (_borderColor != DrawingColor.Transparent)
            using (var path =
GetRoundedRectangle(ClientRectangle, _borderRadius))
                using (var pen = new Pen(_borderColor, 1))
                    pevent.Graphics.DrawPath(pen, path);
        if (!string.IsNullOrEmpty(Text))
            TextRenderer.DrawText(pevent.Graphics, Text,
Font, ClientRectangle, ForeColor, currentBackColor,
TextFormatFlags.HorizontalCenter |
TextFormatFlags.VerticalCenter | TextFormatFlags.SingleLine |
TextFormatFlags.NoPrefix |
TextFormatFlags.PreserveGraphicsClipping);
        using (var path =
GetRoundedRectangle(ClientRectangle, _borderRadius))
            Region = new Region(path);
    }

    protected override void OnPaintBackground(PaintEventArgs
pevent) { }

    private GraphicsPath GetRoundedRectangle(DrawingRectangle
rect, int radius)
    {
        GraphicsPath path = new GraphicsPath();
        if (radius <= 0) { path.AddRectangle(rect); return
path; }

        int diameter = radius * 2;
        path.AddArc(rect.X, rect.Y, diameter, diameter, 180,
90);
    }

```

```

        path.AddArc(rect.Right - diameter, rect.Y, diameter,
diameter, 270, 90);
        path.AddArc(rect.Right - diameter, rect.Bottom -
diameter, diameter, diameter, 0, 90);
        path.AddArc(rect.X, rect.Bottom - diameter, diameter,
diameter, 90, 90);
        path.CloseFigure();
        return path;
    }

    protected override void OnGotFocus(EventArgs e) =>
Invalidate();
    protected override void OnLostFocus(EventArgs e) =>
Invalidate();
    protected override bool ShowFocusCues => false;
}

public class DatabaseManager : IDisposable
{
    private const string DB_FILE = "booklibrary.db";
    private SQLiteConnection _connection;

    public DatabaseManager()
    {
        InitializeDatabase();
    }

    public Dictionary<string, int>
GetReadingStatsByAddedDate(string username)
    {
        var stats = new Dictionary<string, int>();
        using (var cmd = new SQLiteCommand(
            "SELECT AddedDate FROM Favorites WHERE Username =
@u ORDER BY AddedDate;",
            _connection))
        {
            cmd.Parameters.AddWithValue("@u", username);
            using (var reader = cmd.ExecuteReader())
                while (reader.Read())
                {
                    string fullDate = reader.GetString(0);
                    if (!string.IsNullOrEmpty(fullDate) &&
fullDate.Length >= 10)
                    {
                        // Берём только дату (первые 10
СИМВОЛОВ: "ДД.ММ.ГГГГ")
                        string dateOnly = fullDate.Substring(0, 10);

                        if (stats.ContainsKey(dateOnly))
                            stats[dateOnly]++;
                        else
                            stats[dateOnly] = 1;
                    }
                }
        }
    }
}

```

```

    }
    }
    return stats;
}
private void InitializeDatabase()
{
    bool newDb = !File.Exists(DB_FILE);
    _connection = new SQLiteConnection($"Data
Source={DB_FILE};Version=3;Cache=Shared;");
    _connection.Open();

    using (var cmd = new SQLiteCommand("CREATE TABLE IF
NOT EXISTS Users (Id INTEGER PRIMARY KEY AUTOINCREMENT, Username
TEXT UNIQUE NOT NULL, Password TEXT NOT NULL, Theme TEXT DEFAULT
'Светлая', Age INTEGER DEFAULT 18, ContentFilterEnabled INTEGER
DEFAULT 1);", _connection)) cmd.ExecuteNonQuery();
    bool hasAvatar = false;
    using (var cmd = new SQLiteCommand("PRAGMA
table_info(Users);", _connection))
    using (var reader = cmd.ExecuteReader())
    while (reader.Read())
    if (reader.GetString(1) == "Avatar")
hasAvatar = true;
    if (!hasAvatar)
    using (var cmd = new SQLiteCommand("ALTER TABLE
Users ADD COLUMN Avatar BLOB;", _connection))
    cmd.ExecuteNonQuery();
    bool hasAge = false, hasContentFilter = false;
    using (var cmd = new SQLiteCommand("PRAGMA
table_info(Users);", _connection))
    using (var reader = cmd.ExecuteReader())
    while (reader.Read())
    {
        string colName = reader.GetString(1);
        if (colName == "Age") hasAge = true;
        if (colName == "ContentFilterEnabled")
hasContentFilter = true;
    }
    if (!hasAge)
    using (var cmd = new SQLiteCommand("ALTER TABLE
Users ADD COLUMN Age INTEGER DEFAULT 18;", _connection))
    cmd.ExecuteNonQuery();
    if (!hasContentFilter)
    using (var cmd = new SQLiteCommand("ALTER TABLE
Users ADD COLUMN ContentFilterEnabled INTEGER DEFAULT 1;",
_connection))
    cmd.ExecuteNonQuery();
    using (var cmd = new SQLiteCommand("CREATE TABLE IF
NOT EXISTS Favorites (Id INTEGER PRIMARY KEY AUTOINCREMENT,
Username TEXT NOT NULL, Title TEXT NOT NULL, Author TEXT, Year
TEXT, AddedDate TEXT, BookKey TEXT, CoverId TEXT,

```

```

UNIQUE (Username, BookKey));", _connection))
cmd.ExecuteNonQuery();
    bool hasRating = false, hasDescription = false;
    using (var cmd = new SQLiteCommand("PRAGMA
table_info(Favorites);", _connection))
        using (var reader = cmd.ExecuteReader())
            while (reader.Read())
                {
                    string colName = reader.GetString(1);
                    if (colName == "Rating") hasRating = true;
                    if (colName == "Description") hasDescription
= true;
                }
            if (!hasRating)
                using (var cmd = new SQLiteCommand("ALTER TABLE
Favorites ADD COLUMN Rating REAL;", _connection))
                    cmd.ExecuteNonQuery();
            if (!hasDescription)
                using (var cmd = new SQLiteCommand("ALTER TABLE
Favorites ADD COLUMN Description TEXT;", _connection))
                    cmd.ExecuteNonQuery();
            bool hasEmail = false;
            using (var cmd = new SQLiteCommand("PRAGMA
table_info(Users);", _connection))
                using (var reader = cmd.ExecuteReader())
                    {
                        while (reader.Read())
                            {
                                string colName = reader.GetString(1);
                                if (colName == "Email") hasEmail = true;
                            }
                    }
            if (!hasEmail)
                {
                    using (var cmd = new SQLiteCommand("ALTER TABLE
Users ADD COLUMN Email TEXT DEFAULT ''; ", _connection))
                        cmd.ExecuteNonQuery();
                }
            using (var cmd = new SQLiteCommand("CREATE TABLE IF
NOT EXISTS CoverCache (CoverId TEXT PRIMARY KEY, ImageData BLOB,
CachedDate TEXT);", _connection)) cmd.ExecuteNonQuery();
            using (var cmd = new SQLiteCommand("CREATE TABLE IF
NOT EXISTS ViewHistory (Id INTEGER PRIMARY KEY AUTOINCREMENT,
Username TEXT NOT NULL, BookKey TEXT NOT NULL, ViewDate TEXT NOT
NULL);", _connection)) cmd.ExecuteNonQuery();
            using (var cmd = new SQLiteCommand("UPDATE Favorites
SET Year = '-' WHERE Year IS NULL OR Year = ''; ", _connection))
                cmd.ExecuteNonQuery();
            if (newDb)
                using (var cmd = new SQLiteCommand("INSERT INTO
Users (Username, Password, Theme, Age, ContentFilterEnabled)
VALUES ('admin', 'admin123', 'Светлая', 25, 1);", _connection))

```

```

        cmd.ExecuteNonQuery();
    }

    public void SaveAvatar(string username, byte[]
avatarData)
    {
        using (var cmd = new SQLiteCommand("UPDATE Users SET
Avatar = @avatar WHERE Username = @u;", _connection))
        {
            cmd.Parameters.AddWithValue("@avatar",
avatarData ?? (object)DBNull.Value);
            cmd.Parameters.AddWithValue("@u", username);
            cmd.ExecuteNonQuery();
        }
    }

    public byte[] LoadAvatar(string username)
    {
        using (var cmd = new SQLiteCommand("SELECT Avatar
FROM Users WHERE Username = @u;", _connection))
        {
            cmd.Parameters.AddWithValue("@u", username);
            return cmd.ExecuteScalar() as byte[];
        }
    }

    public void UpdateUserEmail(string username, string
email)
    {
        using (var cmd = new SQLiteCommand("UPDATE Users SET
Email = @e WHERE Username = @u;", _connection))
        {
            cmd.Parameters.AddWithValue("@e", email ?? "");
            cmd.Parameters.AddWithValue("@u", username);
            cmd.ExecuteNonQuery();
        }
    }

    public string GetUserEmail(string username)
    {
        using (var cmd = new SQLiteCommand("SELECT Email FROM
Users WHERE Username = @u;", _connection))
        {
            cmd.Parameters.AddWithValue("@u", username);
            var result = cmd.ExecuteScalar();
            return result?.ToString() ?? "";
        }
    }

    public string GetPasswordByEmail(string email)
    {

```

```

        using (var cmd = new SQLiteCommand("SELECT Password
FROM Users WHERE Email = @e;", _connection))
        {
            cmd.Parameters.AddWithValue("@e", email);
            var result = cmd.ExecuteScalar();
            return result?.ToString();
        }
    }

    public string GetUsernameByEmail(string email)
    {
        using (var cmd = new SQLiteCommand("SELECT Username
FROM Users WHERE Email = @e;", _connection))
        {
            cmd.Parameters.AddWithValue("@e", email);
            var result = cmd.ExecuteScalar();
            return result?.ToString();
        }
    }

    public void UpdatePassword(string username, string
newPassword)
    {
        using (var cmd = new SQLiteCommand("UPDATE Users SET
Password = @p WHERE Username = @u;", _connection))
        {
            cmd.Parameters.AddWithValue("@p", newPassword);
            cmd.Parameters.AddWithValue("@u", username);
            cmd.ExecuteNonQuery();
        }
    }

    public void CacheCover(string coverId, byte[] imageData)
    {
        using (var cmd = new SQLiteCommand("INSERT OR REPLACE
INTO CoverCache (CoverId, ImageData, CachedDate) VALUES (@id,
@data, @date);", _connection))
        {
            cmd.Parameters.AddWithValue("@id", coverId);
            cmd.Parameters.AddWithValue("@data", imageData);
            cmd.Parameters.AddWithValue("@date",
DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"));
            cmd.ExecuteNonQuery();
        }
    }

    public byte[] GetCachedCover(string coverId)
    {
        using (var cmd = new SQLiteCommand("SELECT ImageData
FROM CoverCache WHERE CoverId = @id;", _connection))
        {
            cmd.Parameters.AddWithValue("@id", coverId);
            return cmd.ExecuteScalar() as byte[];
        }
    }

```

```

    }
}

public List<MainForm.Book> GetRecommendations(string
username, int count = 5)
{
    var recommendations = new List<MainForm.Book>();
    var userBooks = new HashSet<string>();
    using (var cmd = new SQLiteCommand("SELECT BookKey
FROM Favorites WHERE Username = @u;", _connection))
    {
        cmd.Parameters.AddWithValue("@u", username);
        using (var reader = cmd.ExecuteReader())
            while (reader.Read())
                userBooks.Add(reader.GetString(0));
    }
    if (userBooks.Count < 3) return recommendations;
    var similarUsers = new Dictionary<string, int>();
    var inClause = string.Join(",", userBooks.Select((_,
i) => $"@k{i}"));
    using (var cmd = new SQLiteCommand($"SELECT Username,
COUNT(*) as CommonCount FROM Favorites WHERE BookKey IN
({inClause}) AND Username != @u GROUP BY Username HAVING
CommonCount >= 1 ORDER BY CommonCount DESC LIMIT 10;",
_connection))
    {
        cmd.Parameters.AddWithValue("@u", username);
        for (int i = 0; i < userBooks.Count; i++)
            cmd.Parameters.AddWithValue($"@k{i}", userBooks.ElementAt(i));
        using (var reader = cmd.ExecuteReader())
            while (reader.Read())
                similarUsers[reader.GetString(0)] =
reader.GetInt32(1);
    }
    if (similarUsers.Count == 0) return recommendations;
    foreach (var similarUser in similarUsers.Keys)
    {
        using (var cmd = new SQLiteCommand($"SELECT
DISTINCT Title, Author, Year, BookKey, CoverId, Rating,
Description FROM Favorites WHERE Username = @su AND BookKey NOT
IN ({inClause}) LIMIT 5;", _connection))
        {
            cmd.Parameters.AddWithValue("@su",
similarUser);
            for (int i = 0; i < userBooks.Count; i++)
                cmd.Parameters.AddWithValue($"@k{i}", userBooks.ElementAt(i));
            using (var reader = cmd.ExecuteReader())
            {
                while (reader.Read())
                {
                    try
                    {

```

```

        int? year = null;
        if (!reader.IsDBNull(2) &&
int.TryParse(reader.GetString(2), out int y)) year = y;
        int? coverId = null;
        if (!reader.IsDBNull(4) &&
int.TryParse(reader.GetString(4), out int c)) coverId = c;
        double? rating = null;
        if (!reader.IsDBNull(5)) try
{ rating = reader.GetDouble(5); } catch { rating = null; }
        string description =
reader.IsDBNull(6) ? null : reader.GetString(6);
        recommendations.Add(new
MainForm.Book
        {
            Title = reader.GetString(0),
            AuthorNames =
reader.GetString(1),
            FirstPublishYear = year,
            Key = reader.GetString(3),
            CoverId = coverId,
            Rating = rating,
            Description = description
        });
    }
    catch (Exception ex)
{ Debug.WriteLine($"Ошибка при чтении книги: {ex.Message}"); }
    }
}
return recommendations.GroupBy(b => b.Key).Select(g
=> g.First()).Take(count).ToList();
}

public void SaveViewHistory(string username, string
bookKey)
{
    using (var cmd = new SQLiteCommand("INSERT INTO
ViewHistory (Username, BookKey, ViewDate) VALUES (@u, @k, @d);",
_connection))
    {
        cmd.Parameters.AddWithValue("@u", username);
        cmd.Parameters.AddWithValue("@k", bookKey);
        cmd.Parameters.AddWithValue("@d",
DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"));
        cmd.ExecuteNonQuery();
    }
}

public Dictionary<string, int>
GetReadingStatsByYear(string username)
{

```

```

        var stats = new Dictionary<string, int>();
        using (var cmd = new SQLiteCommand("SELECT Year,
COUNT(*) FROM Favorites WHERE Username = @u GROUP BY Year ORDER
BY Year;", _connection))
        {
            cmd.Parameters.AddWithValue("@u", username);
            using (var reader = cmd.ExecuteReader())
                while (reader.Read())
                {
                    string yearKey = reader.GetString(0);
                    if (!string.IsNullOrEmpty(yearKey) &&
yearKey != "-")
                        stats[yearKey] = reader.GetInt32(1);
                }
        }
        return stats;
    }

    public bool ValidateUser(string username, string
password, out string theme, out int age, out bool
contentFilterEnabled)
    {
        theme = "Светлая"; age = 18; contentFilterEnabled =
true;
        using (var cmd = new SQLiteCommand("SELECT Theme,
Age, ContentFilterEnabled FROM Users WHERE Username = @u AND
Password = @p LIMIT 1;", _connection))
        {
            cmd.Parameters.AddWithValue("@u", username);
            cmd.Parameters.AddWithValue("@p", password);
            using (var reader = cmd.ExecuteReader())
                if (reader.Read())
                {
                    theme = reader.GetString(0);
                    age = reader.GetInt32(1);
                    contentFilterEnabled = reader.GetInt32(2)
== 1;

                    return true;
                }
        }
        return false;
    }

    public (int age, bool contentFilterEnabled)
GetUserSettings(string username)
    {
        using (var cmd = new SQLiteCommand("SELECT Age,
ContentFilterEnabled FROM Users WHERE Username = @u;",
_connection))
        {
            cmd.Parameters.AddWithValue("@u", username);
            using (var reader = cmd.ExecuteReader())

```

```

        if (reader.Read())
            return (reader.GetInt32(0),
reader.GetInt32(1) == 1);
    }
    return (18, true);
}

public bool RegisterUser(string username, string
password, int age, bool contentFilterEnabled = true)
{
    try
    {
        if (age < 18) contentFilterEnabled = true;
        using (var cmd = new SQLiteCommand("INSERT INTO
Users (Username, Password, Theme, Age, ContentFilterEnabled)
VALUES (@u, @p, 'Светлая', @a, @c);", _connection))
        {
            cmd.Parameters.AddWithValue("@u", username);
            cmd.Parameters.AddWithValue("@p", password);
            cmd.Parameters.AddWithValue("@a", age);
            cmd.Parameters.AddWithValue("@c",
contentFilterEnabled ? 1 : 0);
            cmd.ExecuteNonQuery();
            return true;
        }
    }
    catch { return false; }
}

public void UpdateUserTheme(string username, string
theme)
{
    using (var cmd = new SQLiteCommand("UPDATE Users SET
Theme = @t WHERE Username = @u;", _connection))
    {
        cmd.Parameters.AddWithValue("@t", theme);
        cmd.Parameters.AddWithValue("@u", username);
        cmd.ExecuteNonQuery();
    }
}

public void UpdateUserAge(string username, int age)
{
    using (var cmd = new SQLiteCommand("UPDATE Users SET
Age = @a WHERE Username = @u;", _connection))
    {
        cmd.Parameters.AddWithValue("@a", age);
        cmd.Parameters.AddWithValue("@u", username);
        cmd.ExecuteNonQuery();
    }
}

```

```

        public void UpdateContentFilter(string username, bool
enabled)
        {
            using (var cmd = new SQLiteCommand("UPDATE Users SET
ContentFilterEnabled = @e WHERE Username = @u;", _connection))
            {
                cmd.Parameters.AddWithValue("@e", enabled ? 1 :
0);

                cmd.Parameters.AddWithValue("@u", username);
                cmd.ExecuteNonQuery();
            }
        }

        public void SaveFavorite(string username, string title,
string author, string year, string addedDate, string bookKey,
string coverId, double? rating = null, string description = null)
        {
            using (var cmd = new SQLiteCommand("INSERT INTO
Favorites (Username, Title, Author, Year, AddedDate, BookKey,
CoverId, Rating, Description) VALUES (@u, @title, @a, @y, @d, @k,
@c, @r, @desc);", _connection))
            {
                cmd.Parameters.AddWithValue("@u", username);
                cmd.Parameters.AddWithValue("@title", title);
                cmd.Parameters.AddWithValue("@a", author ??
"Неизвестен");
                cmd.Parameters.AddWithValue("@y", year ?? "-");
                cmd.Parameters.AddWithValue("@d", addedDate);
                cmd.Parameters.AddWithValue("@k", bookKey);
                cmd.Parameters.AddWithValue("@c", coverId ?? "");
                cmd.Parameters.AddWithValue("@r",
rating.HasValue ? (object)rating.Value : DBNull.Value);
                cmd.Parameters.AddWithValue("@desc",
description ?? (object)DBNull.Value);
                cmd.ExecuteNonQuery();
            }
        }

        public bool IsBookInFavorites(string username, string
bookKey)
        {
            using (var cmd = new SQLiteCommand("SELECT COUNT(*)
FROM Favorites WHERE Username = @u AND BookKey = @k;",
_connection))
            {
                cmd.Parameters.AddWithValue("@u", username);
                cmd.Parameters.AddWithValue("@k", bookKey);
                return Convert.ToInt64(cmd.ExecuteScalar()) > 0;
            }
        }

```

```

        public List<MainForm.FavoriteBook> LoadFavorites(string
username)
        {
            var list = new List<MainForm.FavoriteBook>();
            using (var cmd = new SQLiteCommand("SELECT Title,
Author, Year, AddedDate, BookKey, CoverId FROM Favorites WHERE
Username = @u ORDER BY AddedDate DESC;", _connection))
            {
                cmd.Parameters.AddWithValue("@u", username);
                using (var reader = cmd.ExecuteReader())
                while (reader.Read())
                {
                    list.Add(new MainForm.FavoriteBook
                    {
                        Username = username,
                        Title = reader.GetString(0),
                        Author = reader.GetString(1),
                        Year = reader.GetString(2),
                        AddedDate = reader.GetString(3),
                        BookKey = reader.GetString(4),
                        CoverId = reader.GetString(5)
                    });
                }
            }
            return list;
        }

        public void RemoveFavorites(string username, List<string>
bookKeys)
        {
            if (bookKeys.Count == 0) return;
            var placeholders = string.Join(",",
bookKeys.Select((_, i) => $"@k{i}"));
            using (var cmd = new SQLiteCommand($"DELETE FROM
Favorites WHERE Username = @u AND BookKey IN ({placeholders});",
_connection))
            {
                cmd.Parameters.AddWithValue("@u", username);
                for (int i = 0; i < bookKeys.Count; i++)
                    cmd.Parameters.AddWithValue($"@k{i}", bookKeys[i]);
                cmd.ExecuteNonQuery();
            }
        }

        public void ClearUserFavorites(string username)
        {
            using (var cmd = new SQLiteCommand("DELETE FROM
Favorites WHERE Username = @u;", _connection))
            {
                cmd.Parameters.AddWithValue("@u", username);
                cmd.ExecuteNonQuery();
            }
        }

```

```

        public int CountUserFavorites(string username)
        {
            using (var cmd = new SQLiteCommand("SELECT COUNT(*)
FROM Favorites WHERE Username = @u;", _connection))
            {
                cmd.Parameters.AddWithValue("@u", username);
                return Convert.ToInt32(cmd.ExecuteScalar());
            }
        }

        public void Dispose()
        {
            _connection?.Close();
            _connection?.Dispose();
        }
    }

    public class LoginForm : Form
    {
        private WinFormsTextBox usernameBox, passwordBox;
        private ModernButton loginButton, registerButton;
        private CheckBox rememberCheckBox;
        private DatabaseManager dbManager;
        private WinFormsLabel forgotPasswordLabel;
        private WinFormsPanel titleBar, mainPanel, inputPanel;
        private WinFormsLabel titleBarLabel, closeBtn,
        minimizeBtn, logoLabel, titleLabel, usernameLabel, passwordLabel;

        public LoginForm()
        {
            dbManager = new DatabaseManager();
            InitializeUI();

            DoubleBuffered = true;
            Resize += LoginForm_Resize;
        }

        private void InitializeUI()
        {
            Text = "BookLibrary - Вход";
            Size = new Size(450, 700);
            MinimumSize = new Size(400, 600);
            StartPosition = FormStartPosition.CenterScreen;
            BackColor = DrawingColor.FromArgb(32, 32, 32);
            ForeColor = DrawingColor.White;
            FormBorderStyle = FormBorderStyle.None;

            titleBar = new WinFormsPanel { Dock = DockStyle.Top,
            Height = 35, BackColor = DrawingColor.FromArgb(32, 32, 32) };
            bool dragging = false; Point startPoint =
            Point.Empty;

```

```

        titleBar.MouseDown += (s, e) => { dragging = true;
startPoint = e.Location; };
        titleBar.MouseMove += (s, e) => { if (dragging)
Location = new Point(Location.X + e.X - startPoint.X, Location.Y
+ e.Y - startPoint.Y); };
        titleBar.MouseUp += (s, e) => dragging = false;
        titleBarLabel = new WinFormsLabel { Text =
"BookLibrary", ForeColor = DrawingColor.White, Font = new
DrawingFont("Segoe UI", 11), Location = new Point(10, 8),
AutoSize = true };
        int btnWidth = 45, btnHeight = 35;
        closeBtn = new WinFormsLabel { Text = "X", Font = new
DrawingFont("Segoe UI", 14), ForeColor = DrawingColor.Gray, Size
= new Size(btnWidth, btnHeight), TextAlign =
ContentAlignment.MiddleCenter, Cursor = Cursors.Hand, BackColor =
DrawingColor.Transparent };
        closeBtn.Click += (s, e) => Application.Exit();
        closeBtn.MouseEnter += (s, e) => closeBtn.BackColor =
DrawingColor.FromArgb(232, 17, 35);
        closeBtn.MouseLeave += (s, e) => closeBtn.BackColor =
DrawingColor.Transparent;
        minimizeBtn = new WinFormsLabel { Text = "-", Font =
new DrawingFont("Segoe UI", 14), ForeColor = DrawingColor.Gray,
Size = new Size(btnWidth, btnHeight), TextAlign =
ContentAlignment.MiddleCenter, Cursor = Cursors.Hand, BackColor =
DrawingColor.Transparent };
        minimizeBtn.Click += (s, e) => WindowState =
FormWindowState.Minimized;
        minimizeBtn.MouseEnter += (s, e) =>
minimizeBtn.BackColor = DrawingColor.FromArgb(60, 60, 60);
        minimizeBtn.MouseLeave += (s, e) =>
minimizeBtn.BackColor = DrawingColor.Transparent;
        titleBar.Controls.Add(titleBarLabel);
        titleBar.Controls.Add(closeBtn);
        titleBar.Controls.Add(minimizeBtn);

        mainPanel = new WinFormsPanel { Dock =
DockStyle.Fill, Padding = new Padding(40), BackColor =
DrawingColor.Transparent };
        logoLabel = new WinFormsLabel { Text = "📖", Font =
new DrawingFont("Segoe UI", 64), Dock = DockStyle.Top, Height =
120, TextAlign = ContentAlignment.MiddleCenter, BackColor =
DrawingColor.Transparent };
        titleLabel = new WinFormsLabel { Text =
"БИБЛИОТЕКАКНИГ", Font = new DrawingFont("Segoe UI", 24,
System.Drawing.FontStyle.Bold), ForeColor =
DrawingColor.FromArgb(0, 151, 251), Dock = DockStyle.Top, Height
= 70, TextAlign = ContentAlignment.MiddleCenter, BackColor =
DrawingColor.Transparent };

```

```

        inputPanel = new WinFormsPanel { Dock =
DockStyle.Fill, Padding = new Padding(10), BackColor =
DrawingColor.Transparent };

        usernameLabel = new WinFormsLabel { Text =
"Имяпользователя", Location = new Point(0, 20), Size = new
Size(340, 25), Font = new DrawingFont("Segoe UI", 10), ForeColor
= DrawingColor.LightGray, Anchor = AnchorStyles.Top |
AnchorStyles.Left | AnchorStyles.Right };
        usernameBox = new WinFormsTextBox { Location = new
Point(0, 48), Size = new Size(340, 45), Font = new
DrawingFont("Segoe UI", 12), BackColor =
DrawingColor.FromArgb(45, 45, 48), ForeColor =
DrawingColor.White, BorderStyle = BorderStyle.None, Anchor =
AnchorStyles.Top | AnchorStyles.Left | AnchorStyles.Right };
        AddPlaceholder(usernameBox, "Введителогин");
        passwordLabel = new WinFormsLabel { Text = "Пароль",
Location = new Point(0, 115), Size = new Size(340, 25), Font =
new DrawingFont("Segoe UI", 10), ForeColor =
DrawingColor.LightGray, Anchor = AnchorStyles.Top |
AnchorStyles.Left | AnchorStyles.Right };
        passwordBox = new WinFormsTextBox { Location = new
Point(0, 143), Size = new Size(340, 45), Font = new
DrawingFont("Segoe UI", 12), BackColor =
DrawingColor.FromArgb(45, 45, 48), ForeColor =
DrawingColor.White, BorderStyle = BorderStyle.None,
UseSystemPasswordChar = true, Anchor = AnchorStyles.Top |
AnchorStyles.Left | AnchorStyles.Right };
        AddPlaceholder(passwordBox, "Введитепароль");
        rememberCheckBox = new CheckBox { Text =
"Запомнитьменя", Location = new Point(0, 200), Size = new
Size(200, 30), Font = new DrawingFont("Segoe UI", 10), ForeColor
= DrawingColor.LightGray, Anchor = AnchorStyles.Top |
AnchorStyles.Left };
        loginButton = new ModernButton { Text = "ВОЙТИ",
Location = new Point(0, 250), Size = new Size(340, 48),
NormalColor = DrawingColor.FromArgb(0, 120, 215), HoverColor =
DrawingColor.FromArgb(0, 150, 250), Anchor = AnchorStyles.Top |
AnchorStyles.Left | AnchorStyles.Right };
        loginButton.Click += LoginButton_Click;
        registerButton = new ModernButton { Text =
"СОЗДАТЬАККАУНТ", Location = new Point(0, 310), Size = new
Size(340, 48), NormalColor = DrawingColor.FromArgb(70, 70, 70),
HoverColor = DrawingColor.FromArgb(90, 90, 90), Anchor =
AnchorStyles.Top | AnchorStyles.Left | AnchorStyles.Right };
        registerButton.Click += RegisterButton_Click;
        forgotPasswordLabel = new WinFormsLabel
{
            Text = "Забылипароль?",
            Font = new DrawingFont("Segoe UI", 9,
System.Drawing.FontStyle.Underline),
            ForeColor = DrawingColor.FromArgb(0, 151, 251),

```

```

        Location = new Point(0, 370),
        Size = new Size(340, 25),
        TextAlign = ContentAlignment.MiddleCenter,
        Cursor = Cursors.Hand,
        Anchor = AnchorStyles.Top | AnchorStyles.Left |
AnchorStyles.Right
    };
    forgotPasswordLabel.Click += (s, e) => {
        var forgotForm = new
ForgotPasswordForm(dbManager);
        forgotForm.ShowDialog(this);
    };
    forgotPasswordLabel.MouseEnter += (s, e) =>
forgotPasswordLabel.ForeColor = DrawingColor.FromArgb(100, 180,
255);
    forgotPasswordLabel.MouseLeave += (s, e) =>
forgotPasswordLabel.ForeColor = DrawingColor.FromArgb(0, 151,
251);

// Добавьте forgotPasswordLabel в список контролов:
inputPanel.Controls.Add(forgotPasswordLabel);

    inputPanel.Controls.AddRange(new Control[]
{ usernameLabel, usernameBox, passwordLabel, passwordBox,
rememberCheckBox, loginButton, registerButton });
    mainPanel.Controls.Add(inputPanel);
    mainPanel.Controls.Add(titleLabel);
    mainPanel.Controls.Add(logoLabel);
    Controls.Add(mainPanel);
    Controls.Add(titleBar);

    usernameBox.Paint += (s, e) =>
DrawTextBoxLine(e.Graphics, usernameBox);
    passwordBox.Paint += (s, e) =>
DrawTextBoxLine(e.Graphics, passwordBox);
    PositionTitleBarButtons();
}

private void LoginForm_Resize(object sender, EventArgs e)
{
    PositionTitleBarButtons();
    int availableWidth = inputPanel.ClientSize.Width -
inputPanel.Padding.Horizontal;
    if (availableWidth > 0)
    {
        usernameBox.Width = passwordBox.Width =
loginButton.Width = registerButton.Width = usernameLabel.Width =
passwordLabel.Width = availableWidth;
    }
    usernameBox.Invalidate();
    passwordBox.Invalidate();
}

```

```

private void PositionTitleBarButtons()
{
    if (closeBtn != null && minimizeBtn != null)
    {
        closeBtn.Location = new Point(ClientSize.Width -
closeBtn.Width, 0);
        minimizeBtn.Location = new Point(ClientSize.Width
- closeBtn.Width - minimizeBtn.Width, 0);
    }
}

protected override void WndProc(ref Message m)
{
    const int WM_NCHITTEST = 0x84;
    const int HTLEFT = 10, HTRIGHT = 11, HTTOP = 12,
HTTOPLEFT = 13, HTTOPRIGHT = 14, HTBOTTOM = 15, HTBOTTOMLEFT =
16, HTBOTTOMRIGHT = 17;
    if (m.Msg == WM_NCHITTEST)
    {
        base.WndProc(ref m);
        Point pt = PointToClient(new
Point(m.LParam.ToInt32() & 0xffff, m.LParam.ToInt32() >> 16));
        Size clientSize = ClientSize;
        int borderWidth = 5;
        if (pt.X <= borderWidth && pt.Y <= borderWidth)
m.Result = (IntPtr)HTTOPLEFT;
        else if (pt.X >= clientSize.Width - borderWidth
&& pt.Y <= borderWidth) m.Result = (IntPtr)HTTOPRIGHT;
        else if (pt.X <= borderWidth && pt.Y >=
clientSize.Height - borderWidth) m.Result = (IntPtr)HTBOTTOMLEFT;
        else if (pt.X >= clientSize.Width - borderWidth
&& pt.Y >= clientSize.Height - borderWidth) m.Result =
(IntPtr)HTBOTTOMRIGHT;
        else if (pt.Y <= borderWidth) m.Result =
(IntPtr)HTTOP;
        else if (pt.Y >= clientSize.Height - borderWidth)
m.Result = (IntPtr)HTBOTTOM;
        else if (pt.X <= borderWidth) m.Result =
(IntPtr)HTLEFT;
        else if (pt.X >= clientSize.Width - borderWidth)
m.Result = (IntPtr)HTRIGHT;
        else return;
    }
    else base.WndProc(ref m);
}

private void DrawTextBoxLine(Graphics g, WinFormsTextBox
tb)
{
    using (var pen = new Pen(DrawingColor.FromArgb(0,
151, 251), 2))

```

```

        g.DrawLine(pen, 0, tb.Height - 2, tb.Width,
tb.Height - 2);
    }

    private void AddPlaceholder(WinFormsTextBox tb, string
placeholder)
    {
        tb.Text = placeholder;
        tb.ForeColor = DrawingColor.Gray;
        tb.Enter += (s, e) => { if (tb.Text == placeholder)
{ tb.Text = ""; tb.ForeColor = DrawingColor.White; } };
        tb.Leave += (s, e) => { if
(string.IsNullOrEmpty(tb.Text)) { tb.Text = placeholder;
tb.ForeColor = DrawingColor.Gray; } };
    }

    private void LoginButton_Click(object sender, EventArgs
e)
    {
        string username = usernameBox.Text.Trim();
        string password = passwordBox.Text;
        if (string.IsNullOrEmpty(username) ||
string.IsNullOrEmpty(password))
        {
            MessageBox.Show("Введите логин и пароль!",
"Внимание", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }
        if (dbManager.ValidateUser(username, password, out
string theme, out int age, out bool contentFilterEnabled))
        {
            Hide();
            var mainForm = new MainForm(username, theme, age,
contentFilterEnabled, dbManager);
            mainForm.FormClosed += (s, args) => Close();
            mainForm.Show();
        }
        else MessageBox.Show("Неверный логин или пароль!",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    private void RegisterButton_Click(object sender,
EventArgs e)
    {
        string username = usernameBox.Text.Trim();
        string password = passwordBox.Text;
        if (username == "Введите логин") username = "";
        if (password == "Введите пароль") password = "";
    }

```

```

        if (string.IsNullOrEmpty(username) ||
string.IsNullOrEmpty(password))
        {

MessageBox.Show("Логинипарольнемогутбытьпустыми!", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }
        if (username.Length < 3)
        { MessageBox.Show("Логиндолженбытьнеменее 3 символов!", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Warning); return; }
        if (password.Length < 4)
        { MessageBox.Show("Парольдолженбытьнеменее 4 символов!",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
return; }

        var registrationForm = new
RegistrationForm(dbManager, username, password);
        registrationForm.ShowDialog(this);
    }
}

public class RegistrationForm : Form
{
    private DatabaseManager dbManager;
    private string username, password;
    private NumericUpDown ageNumeric;
    private CheckBox contentFilterCheckBox;
    private WinFormsTextBox emailBox;
    private ModernButton registerButton, cancelButton;
    private WinFormsLabel infoLabel, ageInfoLabel,
emailValidationLabel;
    private bool isEmailValid = false;

    public RegistrationForm(DatabaseManager db, string user,
string pass)
    {
        dbManager = db;
        username = user;
        password = pass;
        InitializeUI();
    }

    private void InitializeUI()
    {
        Text = "Регистрация - Дополнительная информация";
        Size = new Size(520, 610);
        MinimumSize = new Size(480, 580);
        StartPosition = FormStartPosition.CenterParent;
        BackColor = DrawingColor.FromArgb(32, 32, 32);
        ForeColor = DrawingColor.White;
        FormBorderStyle = FormBorderStyle.None;

        // Заголовок окна

```

```

var titleBar = new WinFormsPanel
{
    Dock = DockStyle.Top,
    Height = 38,
    BackColor = DrawingColor.FromArgb(32, 32, 32)
};

bool dragging = false;
Point startPoint = Point.Empty;
titleBar.MouseDown += (s, e) => { dragging = true;
startPoint = e.Location; };
titleBar.MouseMove += (s, e) => { if (dragging)
Location = new Point(Location.X + e.X - startPoint.X, Location.Y
+ e.Y - startPoint.Y); };
titleBar.MouseUp += (s, e) => dragging = false;

var titleBarLabel = new WinFormsLabel
{
    Text = "Завершениерегистрации",
    ForeColor = DrawingColor.White,
    Font = new DrawingFont("Segoe UI", 11),
    Location = new Point(12, 10),
    AutoSize = true
};

var closeBtn = new WinFormsLabel
{
    Text = "X",
    Font = new DrawingFont("Segoe UI", 14),
    ForeColor = DrawingColor.Gray,
    Size = new Size(45, 38),
    TextAlign = ContentAlignment.MiddleCenter,
    Cursor = Cursors.Hand
};

closeBtn.Click += (s, e) => Close();
closeBtn.MouseEnter += (s, e) => closeBtn.BackColor =
DrawingColor.FromArgb(232, 17, 35);
closeBtn.MouseLeave += (s, e) => closeBtn.BackColor =
DrawingColor.Transparent;

titleBar.Controls.Add(titleBarLabel);
titleBar.Controls.Add(closeBtn);

// Основнаяпанель
var mainPanel = new WinFormsPanel
{
    Dock = DockStyle.Fill,
    AutoScroll = true,
    Padding = new Padding(20)
};

```

```

// Заголовок
var titleLabel = new WinFormsLabel
{
    Text = "📄Дополнительная информация",
    Font = new DrawingFont("Segoe UI", 15,
System.Drawing.FontStyle.Bold),
    ForeColor = DrawingColor.FromArgb(0, 151, 251),
    TextAlign = ContentAlignment.MiddleCenter,
    Dock = DockStyle.Top,
    Height = 45
};

var subtitleLabel = new WinFormsLabel
{
    Text = "Заполните поля ниже для завершения
регистрации",
    Font = new DrawingFont("Segoe UI", 9),
    ForeColor = DrawingColor.Gray,
    TextAlign = ContentAlignment.MiddleCenter,
    Dock = DockStyle.Top,
    Height = 25
};

// Контейнер для полей ввода
var fieldsPanel = new WinFormsPanel
{
    Dock = DockStyle.Top,
    Height = 370,
    Padding = new Padding(10)
};

// Возраст
var ageLabel = new WinFormsLabel
{
    Text = "👑Ваш возраст:",
    Font = new DrawingFont("Segoe UI", 11,
System.Drawing.FontStyle.Bold),
    ForeColor = DrawingColor.White,
    Location = new Point(0, 10),
    Size = new Size(460, 25)
};

ageNumeric = new NumericUpDown
{
    Location = new Point(0, 40),
    Size = new Size(120, 40),
    Font = new DrawingFont("Segoe UI", 12),
    Minimum = 1,
    Maximum = 120,
    Value = 18,
    BackColor = DrawingColor.FromArgb(45, 45, 48),

```

```

        ForeColor = DrawingColor.White,
        BorderStyle = BorderStyle.None,
        TextAlign =
System.Windows.Forms.HorizontalAlignment.Center
    };

    ageInfoLabel = new WinFormsLabel
    {
        Location = new Point(0, 85),
        Size = new Size(460, 30),
        Font = new DrawingFont("Segoe UI", 9),
        ForeColor = DrawingColor.LightGreen,
        Text = "✓Вам 18 летилибольше.
Выможетеотключитьфильтр."
    };

    ageNumeric.ValueChanged += (s, e) =>
    {
        if (ageNumeric.Value < 18)
        {
            contentFilterCheckBox.Checked = true;
            contentFilterCheckBox.Enabled = false;
            ageInfoLabel.Text = "△Вамменьше 18 лет.
Фильтрконтентаобязателен.";
            ageInfoLabel.ForeColor = DrawingColor.Orange;
        }
        else
        {
            contentFilterCheckBox.Enabled = true;
            ageInfoLabel.Text = "✓Вам 18 летилибольше.
Выможетеуправлятьфильтромвпрофиле.";
            ageInfoLabel.ForeColor =
DrawingColor.LightGreen;
        }
    };

    // Фильтрконтента
    var filterLabel = new WinFormsLabel
    {
        Text = "ⓂФильтрконтента:",
        Font = new DrawingFont("Segoe UI", 11,
System.Drawing.FontStyle.Bold),
        ForeColor = DrawingColor.White,
        Location = new Point(0, 125),
        Size = new Size(460, 25)
    };

    contentFilterCheckBox = new CheckBox
    {
        Text = "Скрыватькниги 18+",
        Location = new Point(0, 155),

```

```

        Size = new Size(300, 30),
        Font = new DrawingFont("Segoe UI", 11),
        ForeColor = DrawingColor.LightGray,
        Checked = true
    };

infoLabel = new WinFormsLabel
{
    Text = "Фильтр автоматически скрывает
нежелательный контент при поиске книг",
    Location = new Point(0, 190),
    Size = new Size(460, 30),
    Font = new DrawingFont("Segoe UI", 8),
    ForeColor = DrawingColor.Gray
};

// Email
var emailLabel = new WinFormsLabel
{
    Text = "✉ Email для восстановления пароля:",
    Font = new DrawingFont("Segoe UI", 11,
    System.Drawing.FontStyle.Bold),
    ForeColor = DrawingColor.White,
    Location = new Point(0, 230),
    Size = new Size(460, 25)
};

// Панель для поля email и индикатора валидации
var emailInputPanel = new WinFormsPanel
{
    Location = new Point(0, 260),
    Size = new Size(460, 35),
    BackColor = DrawingColor.FromArgb(45, 45, 48)
};

emailBox = new WinFormsTextBox
{
    Text = "example@mail.com",
    Location = new Point(0, 0),
    Size = new Size(425, 35),
    Font = new DrawingFont("Segoe UI", 11),
    BackColor = DrawingColor.FromArgb(45, 45, 48),
    ForeColor = DrawingColor.Gray,
    BorderStyle = BorderStyle.None
};

// Индикатор валидации email
emailValidationLabel = new WinFormsLabel
{
    Text = "",
    Location = new Point(430, 0),

```

```

        Size = new Size(30, 35),
        Font = new DrawingFont("Segoe UI", 14),
        TextAlign = ContentAlignment.MiddleCenter,
        BackColor = DrawingColor.Transparent
    };

    emailBox.TextChanged += (s, e) => ValidateEmail();
    emailBox.Enter += (s, e) =>
    {
        if (emailBox.Text == "example@mail.com")
        {
            emailBox.Text = "";
            emailBox.ForeColor = DrawingColor.White;
            ValidateEmail();
        }
    };
    emailBox.Leave += (s, e) =>
    {
        if (string.IsNullOrEmpty(emailBox.Text))
        {
            emailBox.Text = "example@mail.com";
            emailBox.ForeColor = DrawingColor.Gray;
            emailValidationLabel.Text = "";
            isEmailValid = false;
        }
        else
        {
            ValidateEmail();
        }
    };

    // Подсказка по email
    var emailHintLabel = new WinFormsLabel
    {
        Text = "Оставьте поле пустым или введите корректный email (например, user@mail.ru)",
        Location = new Point(0, 300),
        Size = new Size(460, 20),
        Font = new DrawingFont("Segoe UI", 8),
        ForeColor = DrawingColor.Gray
    };

    emailInputPanel.Controls.Add(emailBox);
    emailInputPanel.Controls.Add(emailValidationLabel);

    // Инициализация состояния
    if (ageNumeric.Value < 18)
    {
        contentFilterCheckBox.Checked = true;
        contentFilterCheckBox.Enabled = false;
    }

```

```

        ageInfoLabel.Text = "△Вамменьше 18 лет.
Фильтрконтентаобязателен.";
ageInfoLabel.ForeColor = DrawingColor.Orange;
    }
else
    {
contentFilterCheckBox.Enabled = true;
    }

    // Добавляемвсеэлементывпанельполей
fieldsPanel.Controls.AddRange(new Control[]
    {
        ageLabel, ageNumeric, ageInfoLabel,
        filterLabel, contentFilterCheckBox, infoLabel,
        emailLabel, emailInputPanel, emailHintLabel
    });

    // Панельскнопками
var buttonPanel = new WinFormsPanel
{
    Dock = DockStyle.Top,
    Height = 55,
    Padding = new Padding(0, 10, 0, 0)
};

    registerButton = new ModernButton
{
    Text = "✓ЗАВЕРШИТЬРЕГИСТРАЦИЮ",
    Size = new Size(250, 45),
    NormalColor = DrawingColor.FromArgb(46, 204, 113),
        HoverColor = DrawingColor.FromArgb(60, 220, 130),
        Font = new DrawingFont("Segoe UI", 10,
System.Drawing.FontStyle.Bold)
    };
    registerButton.Click += RegisterButton_Click;

    cancelButton = new ModernButton
{
    Text = "✗ОТМЕНА",
    Size = new Size(120, 45),
    NormalColor = DrawingColor.FromArgb(100, 100,
100),
        Font = new DrawingFont("Segoe UI", 10)
    };
    cancelButton.Click += (s, e) => Close();

    buttonPanel.Controls.Add(registerButton);
    buttonPanel.Controls.Add(cancelButton);

    // Компоновка
mainPanel.Controls.Add(buttonPanel);

```

```

mainPanel.Controls.Add(fieldsPanel);
mainPanel.Controls.Add(subtitleLabel);
mainPanel.Controls.Add(titleLabel);

Controls.Add(mainPanel);
Controls.Add(titleBar);

Resize += (s, e) => {
    closeBtn.Location = new Point(ClientSize.Width -
48, 0);

    // Центрируем кнопки
    registerButton.Left =
(buttonPanel.ClientSize.Width - registerButton.Width -
cancelButton.Width - 10) / 2;
    cancelButton.Left = registerButton.Right + 10;

    // Обновляем ширину полей
    int availableWidth = fieldsPanel.ClientSize.Width
- 20;

    ageLabel.Width = availableWidth;
    filterLabel.Width = availableWidth;
    emailLabel.Width = availableWidth;
    ageInfoLabel.Width = availableWidth;
    infoLabel.Width = availableWidth;
    emailInputPanel.Width = availableWidth;
    emailHintLabel.Width = availableWidth;
    emailBox.Width = availableWidth - 30;
    emailValidationLabel.Left = availableWidth - 30;
};

// Первоначальное центрирование
Shown += (s, e) => {
    registerButton.Left =
(buttonPanel.ClientSize.Width - registerButton.Width -
cancelButton.Width - 10) / 2;
    cancelButton.Left = registerButton.Right + 10;
};
}

private void ValidateEmail()
{
    string email = emailBox.Text.Trim();

    // Если поле пустое или содержит плейсхолдер
    if (string.IsNullOrEmpty(email) || email == "example@mail.com")
    {
        emailValidationLabel.Text = "";
        isEmailValid = false;
        UpdateEmailFieldBorder(DrawingColor.FromArgb(0,
151, 251));

        return;
    }
}

```

```

}

// Проверка на корректность email
if (IsValidEmail(email))
{
    emailValidationLabel.Text = "✓";
    emailValidationLabel.ForeColor =
DrawingColor.LightGreen;
    isEmailValid = true;
    UpdateEmailFieldBorder(DrawingColor.LightGreen);
}
else
{
    emailValidationLabel.Text = "✗";
    emailValidationLabel.ForeColor =
DrawingColor.FromArgb(255, 100, 100);
    isEmailValid = false;
    UpdateEmailFieldBorder(DrawingColor.FromArgb(255,
100, 100));
}

private void UpdateEmailFieldBorder(DrawingColor color)
{
    // Перерисовываем нижнюю линию поля email
    emailBox.Invalidate();
    emailBox.Paint -= EmailBox_Paint;
    emailBox.Paint += EmailBox_Paint;

    void EmailBox_Paint(object sender, PaintEventArgs e)
    {
        using (var pen = new Pen(color, 2))
            e.Graphics.DrawLine(pen, 0, emailBox.Height -
2, emailBox.Width, emailBox.Height - 2);
    }
}

private bool IsValidEmail(string email)
{
    if (string.IsNullOrEmpty(email))
        return false;

    try
    {
        // Используем System.Net.Mail.MailAddress
для проверки
        var addr = new
System.Net.Mail.MailAddress(email);
        return addr.Address == email;
    }
    catch

```

```

        {
            // Дополнительная проверка регулярным выражением
            return
System.Text.RegularExpressions.Regex.IsMatch(email,
            @"^[^@\s]+@^[^@\s]+\.[^@\s]+$",

System.Text.RegularExpressions.RegexOptions.IgnoreCase);
        }
    }

    private void RegisterButton_Click(object sender,
EventArgs e)
    {
        int age = (int)ageNumeric.Value;
        bool contentFilter = contentFilterCheckBox.Checked;

        if (age < 18 && !contentFilter)
            contentFilter = true;

        string email = emailBox.Text.Trim();
        if (email == "example@mail.com")
email = "";

        // Проверяем email если он введен
        if (!string.IsNullOrEmpty(email) && !IsValidEmail(email))
        {
            MessageBox.Show(
                "Введите корректный email адрес или оставьте
поле пустым.",
                "Некорректный email",
                MessageBoxButtons.OK,
                MessageBoxIcon.Warning);
            emailBox.Focus();
            return;
        }

        if (dbManager.RegisterUser(username, password, age,
contentFilter))
        {
            if (!string.IsNullOrEmpty(email))
            {
                dbManager.UpdateUserEmail(username, email);
            }

            MessageBox.Show(
                $"Аккаунт успешно создан!\n\n" +
                $"👤 Логин: {username}\n" +
                $"🎂 Возраст: {age} лет\n" +
                $"🔞 Фильтр контента: {(contentFilter ? "Включен" :
"Отключен")}\n" +

```

```

$"📧 Email: {(string.IsNullOrEmpty(email) ? "не указан" :
email)}\n\n" +
                (age < 18 ? "🔞 Вам меньше 18 лет. Фильтр
контента будет активен всегда." : "Теперь вы можете войти в
систему."),
                "✅ Успешная регистрация",
                MessageBoxButtons.OK,
                MessageBoxIcon.Information);
            Close();
        }
        else
        {
            MessageBox.Show("❌ Пользователь с таким логином
уже существует!", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
        }
    }
}

public class ForgotPasswordForm : Form
{
    private WinFormsTextBox emailBox;
    private ModernButton sendButton, cancelButton;
    private WinFormsLabel infoLabel, titleLabel;
    private DatabaseManager dbManager;

    public ForgotPasswordForm(DatabaseManager db)
    {
        dbManager = db;
        InitializeUI();
    }

    private void InitializeUI()
    {
        Text = "Восстановление пароля";
        Size = new Size(450, 350);
        StartPosition = FormStartPosition.CenterParent;
        BackColor = DrawingColor.FromArgb(32, 32, 32);
        ForeColor = DrawingColor.White;
        FormBorderStyle = FormBorderStyle.None;

        var titleBar = new WinFormsPanel { Dock =
DockStyle.Top, Height = 35, BackColor = DrawingColor.FromArgb(32,
32, 32) };
        bool dragging = false; Point startPoint =
Point.Empty;
        titleBar.MouseDown += (s, e) => { dragging = true;
startPoint = e.Location; };
        titleBar.MouseMove += (s, e) => { if (dragging)
Location = new Point(Location.X + e.X - startPoint.X, Location.Y
+ e.Y - startPoint.Y); };
    }
}

```

```

        titleBar.MouseUp += (s, e) => dragging = false;

        var titleBarLabel = new WinFormsLabel { Text =
"Восстановление пароля", ForeColor = DrawingColor.White, Font =
new DrawingFont("Segoe UI", 11), Location = new Point(10, 8),
AutoSize = true };

        var closeBtn = new WinFormsLabel { Text = "X", Font =
new DrawingFont("Segoe UI", 14), ForeColor = DrawingColor.Gray,
Size = new Size(45, 35), TextAlign =
ContentAlignment.MiddleCenter, Cursor = Cursors.Hand, Location =
new Point(405, 0) };

        closeBtn.Click += (s, e) => Close();
        closeBtn.MouseEnter += (s, e) => closeBtn.BackColor =
DrawingColor.FromArgb(232, 17, 35);
        closeBtn.MouseLeave += (s, e) => closeBtn.BackColor =
DrawingColor.Transparent;
        titleBar.Controls.Add(titleBarLabel);
        titleBar.Controls.Add(closeBtn);

        titleLabel = new WinFormsLabel
        {
            Text = "Введите email для восстановления пароля",
            Font = new DrawingFont("Segoe UI", 12,
System.Drawing.FontStyle.Bold),
            ForeColor = DrawingColor.FromArgb(0, 151, 251),
            Location = new Point(30, 60),
            Size = new Size(390, 30),
            TextAlign = ContentAlignment.MiddleCenter
        };

        emailBox = new WinFormsTextBox
        {
            Location = new Point(30, 110),
            Size = new Size(390, 45),
            Font = new DrawingFont("Segoe UI", 12),
            BackColor = DrawingColor.FromArgb(45, 45, 48),
            ForeColor = DrawingColor.White,
            BorderStyle = BorderStyle.None
        };
        emailBox.Paint += (s, e) => {
            using (var pen = new Pen(DrawingColor.FromArgb(0,
151, 251), 2))
                e.Graphics.DrawLine(pen, 0, emailBox.Height -
2, emailBox.Width, emailBox.Height - 2);
        };

        infoLabel = new WinFormsLabel
        {
            Text = "",
            Font = new DrawingFont("Segoe UI", 9),
            ForeColor = DrawingColor.Gray,

```

```

        Location = new Point(30, 170),
        Size = new Size(390, 40),
        TextAlign = ContentAlignment.MiddleLeft
    };

    sendButton = new ModernButton
    {
        Text = "✉️ОТПРАВИТЬНАEMAIL",
        Location = new Point(30, 230),
        Size = new Size(250, 45),
        NormalColor = DrawingColor.FromArgb(0, 120, 215),
        HoverColor = DrawingColor.FromArgb(0, 150, 250)
    };
    sendButton.Click += SendButton_Click;

    cancelButton = new ModernButton
    {
        Text = "✖ОТМЕЧА",
        Location = new Point(290, 230),
        Size = new Size(130, 45),
        NormalColor = DrawingColor.FromArgb(100, 100,
100)
    };
    cancelButton.Click += (s, e) => Close();

    Controls.AddRange(new Control[] { titleBar,
titleLabel, emailBox, infoLabel, sendButton, cancelButton });
    Resize += (s, e) => closeBtn.Location = new
Point(ClientSize.Width - 45, 0);
    }

    private void SendButton_Click(object sender, EventArgs e)
    {
        string email = emailBox.Text.Trim();
        if (string.IsNullOrEmpty(email)
|| !email.Contains("@"))
        {
            MessageBox.Show("Введитекорректный email!",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }

        string username =
dbManager.GetUsernameByEmail(email);
        if (string.IsNullOrEmpty(username))
        {
            infoLabel.Text = "✖ Email ненайденвсистеме";
            infoLabel.ForeColor = DrawingColor.FromArgb(255,
100, 100);
            return;
        }
    }

```

```

        string password =
dbManager.GetPasswordByEmail(email);
        string newPassword =
Guid.NewGuid().ToString().Substring(0, 8);
        dbManager.UpdatePassword(username, newPassword);

        string tempFile = Path.Combine(Path.GetTempPath(),
$password_recovery_{username}.txt");
        File.WriteAllText(tempFile,
            $"Восстановление пароля BookLibrary\n\n" +
            $"Логин: {username}\n" +
            $"Новый пароль: {newPassword}\n\n" +
            $"Рекомендуется сменить пароль после входа.\n" +
            $"Создано: {DateTime.Now:dd.MM.yyyy HH:mm}");

try
    {
        Process.Start(new ProcessStartInfo(tempFile)
        { UseShellExecute = true });
        infoLabel.Text = "✓ Новый пароль сохранён в файл";
        infoLabel.ForeColor = DrawingColor.FromArgb(100, 255, 100);
        MessageBox.Show(
            $"Новый пароль для пользователя {username}\n" +
            $"сохранён в файл:\n{tempFile}\n\n" +
            $"Отправка email не настроена.\n" +
            $"Пароль сохранён локально.",
            "Восстановление пароля",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
    catch
    {
        infoLabel.Text = $"✓ Пароль сброшен. Логин:
{username}";
        infoLabel.ForeColor = DrawingColor.FromArgb(100,
255, 100);
    }
}

public class MainForm : Form
{
    private TabControl mainTabControl;
    private ListView searchResultsListView,
favoritesListView;
    private RichTextBox bookDetailsTextBox;
    private WinFormsPictureBox bookCoverPicture;
    private WinFormsTextBox searchBox;
    private ModernButton searchButton, addToFavoritesButton,
removeFromFavoritesButton, clearFavoritesButton, logoutButton,
exportPdfButton;

```

```

        private WinFormsLabel welcomeLabel, statusLabel,
favoritesCountLabel, currentThemeLabel, userAgeLabel,
filterStatusLabel;
        private WinFormsProgressBar progressBar;
        private FlowLayoutPanel recommendationsPanel;
        private bool isFullScreenOpen = false;
        private PictureBox avatarPictureBox;
        private Button changeAvatarButton;
        private HttpClient httpClient;
        private string currentUsername;
        private string currentTheme;
        private int userAge;
        private bool contentFilterEnabled;
        private List<Book> currentBooks = new List<Book>();
        private DatabaseManager dbManager;
        private bool recommendationsLoaded = false;

        private static readonly HashSet<string> AdultKeywords =
new HashSet<string>(StringComparer.OrdinalIgnoreCase)
        {
            "erotic", "erotica", "sex", "sexual", "porn",
            "porno", "pornography", "adult", "xxx", "nude", "naked",
            "explicit", "bdsm", "fetish",
            "kama sutra", "tantra", "sensual", "seduction",
            "orgasm", "intercourse", "masturbation", "prostitute", "escort",
            "эротика", "секс", "сексуальный", "порно", "порнография", "18+",
            "18 плюс", "интим", "интимный", "разврат", "развратный",
            "голый", "обнаженный", "обнажённый", "вульгарный",
            "пошлый", "камасутра", "тантра", "соблазнение", "оргазм",
            "вульва",
            "пенис", "вагина", "грудь", "обнаженная", "ню",
            "horny", "sexy", "lingerie", "strip", "stripper", "nudist",
            "playboy", "penthouse", "hustler", "mature", "taboo"
        };

        private readonly Dictionary<string, ColorScheme> themes =
new Dictionary<string, ColorScheme>
        {
            ["Светлая"] = new ColorScheme { Background =
DrawingColor.White, Foreground = DrawingColor.Black,
ControlBackground = DrawingColor.FromArgb(245, 245, 245),
AccentColor = DrawingColor.FromArgb(0, 120, 215), SecondaryColor
= DrawingColor.FromArgb(70, 130, 180) },
            ["Темная"] = new ColorScheme { Background =
DrawingColor.FromArgb(32, 32, 32), Foreground =
DrawingColor.White, ControlBackground = DrawingColor.FromArgb(45,
45, 48), AccentColor = DrawingColor.FromArgb(0, 151, 251),
SecondaryColor = DrawingColor.FromArgb(100, 160, 210) },
            ["Лесная"] = new ColorScheme { Background =
DrawingColor.FromArgb(240, 248, 240), Foreground =
DrawingColor.FromArgb(30, 60, 30), ControlBackground =
DrawingColor.FromArgb(220, 235, 220), AccentColor =

```

```

DrawingColor.FromArgb(76, 175, 80), SecondaryColor =
DrawingColor.FromArgb(106, 195, 110) }
};

private const string OPEN_LIBRARY_SEARCH_URL =
"https://openlibrary.org/search.json";
private const string OPEN_LIBRARY_BOOK_URL =
"https://openlibrary.org";
private const string COVER_IMAGE_URL =
"https://covers.openlibrary.org/b";

public MainForm(string username, string theme, int age,
bool contentFilter, DatabaseManager db)
{
    currentUsername = username;
    currentTheme = theme;
    userAge = age;
    dbManager = db;
    contentFilterEnabled = userAge < 18 ? true :
contentFilter;
    if (userAge < 18)
dbManager.UpdateContentFilter(currentUsername, true);
    InitializeHttpClient();
    InitializeUI();
    LoadAvatar();
    ApplyTheme();
    DoubleBuffered = true;
}

private void RefreshStatsPlot(FormsPlot plotView)
{
    plotView.Plot.Clear();

    var stats =
dbManager.GetReadingStatsByAddedDate(currentUsername);

    if (stats.Count > 0)
    {
        var sortedStats = stats.OrderBy(kv =>
        {
            if (DateTime.TryParseExact(kv.Key,
"dd.MM.yyyy", null, System.Globalization.DateTimeStyles.None, out
DateTime date))
                return date;
            return DateTime.MinValue;
        }).ToList();

        if (sortedStats.Count > 0)
        {
            var dates = new List<string>();
            var counts = new List<double>();

```

```

        foreach (var kv in sortedStats)
        {
            dates.Add(kv.Key);
            counts.Add(kv.Value);
        }

        double[] positions = Enumerable.Range(0,
counts.Count).Select(x => (double)x).ToArray();
        double[] values = counts.ToArray();

        var bar = plotView.Plot.Add.Bars(positions,
values);

// Настройка подписей в зависимости от количества дат
        if (dates.Count <= 7)
        {
            // Мало дат — показываем все
горизонтально
            plotView.Plot.Axes.Bottom.SetTicks(positions, dates.ToArray());

            plotView.Plot.Axes.Bottom.TickLabelStyle.Rotation = 0;

            plotView.Plot.Axes.Bottom.TickLabelStyle.FontSize = 12;

            plotView.Plot.Axes.Bottom.TickLabelStyle.Alignment =
Alignment.MiddleCenter;
        }

        else if (dates.Count <= 15)
        {
            // Среднее количество — показываем все
под углом 45°
            plotView.Plot.Axes.Bottom.SetTicks(positions, dates.ToArray());

            plotView.Plot.Axes.Bottom.TickLabelStyle.Rotation = 45;

            plotView.Plot.Axes.Bottom.TickLabelStyle.FontSize = 10;

            plotView.Plot.Axes.Bottom.TickLabelStyle.Alignment =
Alignment.UpperLeft;
        }

        else
        {
            // Много дат — показываем с шагом,
вертикально
            int step = dates.Count / 15;

            var filteredTicks = new List<double>();
            var filteredLabels = new List<string>();
            for (int i = 0; i < positions.Length; i
+= step)

                {
                    filteredTicks.Add(positions[i]);
                    filteredLabels.Add(dates[i]);
                }
        }
    }
}

```

```

    }

    // Добавляем последнюю дату, если её нет
    if (!filteredTicks.Contains(positions.Last()))
    {
        filteredTicks.Add(positions.Last());
        filteredLabels.Add(dates.Last());
    }

    plotView.Plot.Axes.Bottom.SetTicks(filteredTicks.ToArray(),
    filteredLabels.ToArray());

    plotView.Plot.Axes.Bottom.TickLabelStyle.Rotation = 90;

    plotView.Plot.Axes.Bottom.TickLabelStyle.FontSize = 8;

    plotView.Plot.Axes.Bottom.TickLabelStyle.Alignment =
    Alignment.UpperRight;
    }

    plotView.Plot.Title($"Всего добавлено книг:
    {stats.Values.Sum()}");
    plotView.Plot.XLabel("Дата добавления
    (ДД.ММ.ГГГГ)");
    plotView.Plot.YLabel("Количество книг");

    // Фиксированные целые числа по оси Y
    int maxY = (int)values.Max() + 1;
    double[] yTicks = Enumerable.Range(0, maxY +
    1).Select(x => (double)x).ToArray();
    string[] yLabels = Enumerable.Range(0, maxY +
    1).Select(x => x.ToString()).ToArray();
    plotView.Plot.Axes.Left.SetTicks(yTicks,
    yLabels);

    // Отключаем увеличение и привязываем оси к данным
    plotView.Plot.Axes.SetLimits(
        -0.5, positions.Length - 0.5, // X: от
    первого до последнего столбца
        0, maxY // Y: от
    0 до максимума
    );

    plotView.Refresh();
    }
    else
    {
        plotView.Plot.Title("Нет данных для
    отображения");
        plotView.Refresh();
    }
    }
}

```

```

        else
        {
            plotView.Plot.Title("Нет данных для
отображения");
plotView.Refresh();
        }
    }
    private void RefreshStatsGraph()
    {
        try
        {
            // Ищем график по имени на вкладке профиля
var profileTab = mainTabControl.TabPages.Cast<TabPage>()
                .FirstOrDefault(t =>
t.Text.Contains("Профиль"));

            if (profileTab != null)
            {
                var plotView = FindControlByName(profileTab,
"statsPlot") as FormsPlot;
                if (plotView != null)
                {
                    RefreshStatsPlot(plotView);
                }
            }
        }
        catch (Exception ex)
        {
            Debug.WriteLine($"Ошибкаобновленияграфика:
{ex.Message}");
        }
    }

    private Control FindControlByName(Control parent, string
name)
    {
        if (parent.Name == name)
            return parent;

        foreach (Control child in parent.Controls)
        {
            Control found = FindControlByName(child, name);
            if (found != null)
                return found;
        }

        return null;
    }

    private void UpdateProfileEmail()
    {
        try

```

```

        {
            var profileTab =
mainTabControl.TabPages.Cast<TabPage>()
                .FirstOrDefault(t =>
t.Text.Contains("Профиль"));

            if (profileTab != null)
            {
                UpdateEmailLabels(profileTab);
            }
        }
        catch (Exception ex)
        {
            Debug.WriteLine($"Ошибкаобновления email:
{ex.Message}");
        }
    }

    private void UpdateEmailLabels(Control parent)
    {
        foreach (Control c in parent.Controls)
        {
            if (c is WinFormsLabel lbl &&
lbl.Text.StartsWith("✉ Email:"))
            {
                lbl.Text = $"✉ Email:
{dbManager.GetUserEmail(currentUsername)}";
                return;
            }

            if (c.HasChildren)
                UpdateEmailLabels(c);
        }
    }

    private void InitializeHttpClient()
    {
        httpClient = new HttpClient { Timeout =
        TimeSpan.FromSeconds(30) };
        httpClient.DefaultRequestHeaders.Add("User-Agent",
        "BookLibraryApp/3.0");
    }

    private void BookDetailsTextBox_LinkClicked(object
sender, LinkClickedEventArgs e)
    {
        try
        {
            if (!string.IsNullOrEmpty(e.LinkText))
            {
                Process.Start(new
                ProcessStartInfo(e.LinkText) { UseShellExecute = true });
            }
        }
    }

```

```

    }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка при открытии ссылки:
{ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}
private void ShowAvatarFullScreen()
{
    if (isFullScreenOpen || avatarPictureBox.Image ==
null) return;
    byte[] originalAvatarData =
dbManager.LoadAvatar(currentUsername);
    if (originalAvatarData == null ||
originalAvatarData.Length == 0)
    {
        MessageBox.Show("Не удалось загрузить изображение", "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error); return; }
    DrawingImage originalImage;
    using (var ms = new MemoryStream(originalAvatarData))
    originalImage = DrawingImage.FromStream(ms);
    Form fullScreenForm = new Form { FormBorderStyle =
FormBorderStyle.None, WindowState = FormWindowState.Maximized,
BackColor = DrawingColor.Black, Opacity = 0, KeyPreview = true,
ShowInTaskbar = false, StartPosition =
FormStartPosition.CenterScreen };
    Panel scrollPanel = new Panel { Dock =
DockStyle.Fill, AutoScroll = true, BackColor =
DrawingColor.Black };
    PictureBox fullPic = new PictureBox { Image =
originalImage, SizeMode = PictureBoxSizeMode.AutoSize, BackColor
= DrawingColor.Black, Cursor = Cursors.Hand };
    scrollPanel.Controls.Add(fullPic);
    Button closeButton = new Button { Text = "X", Size =
new Size(50, 50), Location = new Point(20, 20), FlatStyle =
FlatStyle.Flat, BackColor = DrawingColor.FromArgb(200, 0, 0, 0),
ForeColor = DrawingColor.White, Font = new DrawingFont("Segoe
UI", 20, System.Drawing.FontStyle.Bold), Cursor = Cursors.Hand };
    closeButton.FlatAppearance.BorderSize = 0;
    closeButton.Click += (s, e) => { isFullScreenOpen =
false; fullScreenForm.Close(); };
    Button downloadButton = new Button { Text = "📁",
Size = new Size(50, 50), Location = new Point(80, 20), FlatStyle
= FlatStyle.Flat, BackColor = DrawingColor.FromArgb(200, 0, 0,
0), ForeColor = DrawingColor.White, Font = new DrawingFont("Segoe
UI", 20), Cursor = Cursors.Hand };
    downloadButton.FlatAppearance.BorderSize = 0;
    downloadButton.Click += (s, e) => {

```

```

        using (SaveFileDialog sfd = new SaveFileDialog()
{ Filter = "PNG Image|*.png|JPEG Image|*.jpg", FileName =
$"avatar_{currentUsername}_{DateTime.Now:yyyyMMdd_HH:mm:ss}.png" })
        if (sfd.ShowDialog() == DialogResult.OK)
            try { originalImage.Save(sfd.FileName,
System.Drawing.Imaging.ImageFormat.Png);
MessageBox.Show("Оригиналаватаркисохранён!", "Успех",
MessageBoxButtons.OK, MessageBoxIcon.Information); }
            catch (Exception ex)
{ MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error); }
    };
    Button fitButton = new Button { Text = "[F]", Size =
new Size(50, 50), Location = new Point(140, 20), FlatStyle =
FlatStyle.Flat, BackColor = DrawingColor.FromArgb(200, 0, 0, 0),
ForeColor = DrawingColor.White, Font = new DrawingFont("Segoe
UI", 20), Cursor = Cursors.Hand };
    fitButton.FlatAppearance.BorderSize = 0;
    fitButton.Click += (s, e) => { if (fullPic.SizeMode
== PictureBoxSizeMode.AutoSize) { fullPic.SizeMode =
PictureBoxSizeMode.Zoom; fullPic.Dock = DockStyle.Fill;
fitButton.Text = "Q"; } else { fullPic.SizeMode =
PictureBoxSizeMode.AutoSize; fullPic.Dock = DockStyle.None;
fitButton.Text = "[F]"; } };
    System.Windows.Forms.Label hintLabel = new
System.Windows.Forms.Label { Text = "ESC - закрыть |  -
сохранить оригинал | [F] - подогнать под экран", ForeColor =
DrawingColor.FromArgb(150, 255, 255, 255), BackColor =
DrawingColor.Transparent, Font = new DrawingFont("Segoe UI", 12),
AutoSize = true, Location = new Point(20,
fullScreenForm.ClientSize.Height - 50), Anchor =
AnchorStyles.Bottom | AnchorStyles.Left };
    fullScreenForm.Controls.Add(scrollPanel);
    fullScreenForm.Controls.Add(closeButton);
    fullScreenForm.Controls.Add(downloadButton);
    fullScreenForm.Controls.Add(fitButton);
    fullScreenForm.Controls.Add(hintLabel);
    closeButton.BringToFront();
downloadButton.BringToFront(); fitButton.BringToFront();
hintLabel.BringToFront();
    fullScreenForm.KeyDown += (s, e) => { if (e.KeyCode
== Keys.Escape) { isFullScreenOpen = false;
fullScreenForm.Close(); } };
    fullScreenForm.FormClosed += (s, e) =>
{ isFullScreenOpen = false; originalImage.Dispose(); };
    isFullScreenOpen = true;
    fullScreenForm.Shown += async (s, e) => { for (int i
= 0; i <= 10; i++) { fullScreenForm.Opacity = i / 10.0; await
Task.Delay(10); } };
    fullScreenForm.ShowDialog();
}

```

```

private void InitializeUI()
{
    Text = $"BookLibrary Pro – {currentUsername}
(Bospact: {userAge})";
    Size = new Size(1400, 900);
    MinimumSize = new Size(1000, 700);
    StartPosition = FormStartPosition.CenterScreen;
    Font = new DrawingFont("Segoe UI", 10);
    FormBorderStyle = FormBorderStyle.None;

    var titleBar = new WinFormsPanel { Dock =
DockStyle.Top, Height = 35, BackColor = DrawingColor.FromArgb(32,
32, 32) };
    bool dragging = false; Point startPoint =
Point.Empty;
    titleBar.MouseDown += (s, e) => { dragging = true;
startPoint = e.Location; };
    titleBar.MouseMove += (s, e) => { if (dragging)
Location = new Point(Location.X + e.X - startPoint.X, Location.Y
+ e.Y - startPoint.Y); };
    titleBar.MouseUp += (s, e) => dragging = false;
    var titleBarLabel = new WinFormsLabel { Text = "📖
BookLibrary Pro", ForeColor = DrawingColor.White, Font = new
DrawingFont("Segoe UI", 11), Location = new Point(10, 8),
AutoSize = true };
    int btnWidth = 45, btnHeight = 35, rightOffset = 0;
    var closeBtn = new WinFormsLabel { Text = "X", Font =
new DrawingFont("Segoe UI", 14), ForeColor = DrawingColor.Gray,
Size = new Size(btnWidth, btnHeight), TextAlign =
ContentAlignment.MiddleCenter, Cursor = Cursors.Hand, Location =
new Point(Width - btnWidth - rightOffset, 0) };
    closeBtn.Click += (s, e) => Close();
    closeBtn.MouseEnter += (s, e) => closeBtn.BackColor =
DrawingColor.FromArgb(232, 17, 35);
    closeBtn.MouseLeave += (s, e) => closeBtn.BackColor =
DrawingColor.Transparent;
    rightOffset += btnWidth;
    var maximizeBtn = new WinFormsLabel { Text = "□",
Font = new DrawingFont("Segoe UI", 14), ForeColor =
DrawingColor.Gray, Size = new Size(btnWidth, btnHeight),
TextAlign = ContentAlignment.MiddleCenter, Cursor = Cursors.Hand,
Location = new Point(Width - btnWidth - rightOffset, 0) };
    maximizeBtn.Click += (s, e) => { if (WindowState ==
FormWindowState.Normal) { WindowState =
FormWindowState.Maximized; maximizeBtn.Text = "□"; } else
{ WindowState = FormWindowState.Normal; maximizeBtn.Text =
"□"; } };
    maximizeBtn.MouseEnter += (s, e) =>
maximizeBtn.BackColor = DrawingColor.FromArgb(60, 60, 60);

```

```

        maximizeBtn.MouseLeave += (s, e) =>
maximizeBtn.BackColor = DrawingColor.Transparent;
        rightOffset += btnWidth;
        var minimizeBtn = new WinFormsLabel { Text = "-",
Font = new DrawingFont("Segoe UI", 14), ForeColor =
DrawingColor.Gray, Size = new Size(btnWidth, btnHeight),
TextAlign = ContentAlignment.MiddleCenter, Cursor = Cursors.Hand,
Location = new Point(Width - btnWidth - rightOffset, 0) };
        minimizeBtn.Click += (s, e) => WindowState =
FormWindowState.Minimized;
        minimizeBtn.MouseEnter += (s, e) =>
minimizeBtn.BackColor = DrawingColor.FromArgb(60, 60, 60);
        minimizeBtn.MouseLeave += (s, e) =>
minimizeBtn.BackColor = DrawingColor.Transparent;
        titleBar.Controls.Add(titleBarLabel);
        titleBar.Controls.Add(closeBtn);
        titleBar.Controls.Add(maximizeBtn);
        titleBar.Controls.Add(minimizeBtn);
        Resize += (s, e) => { closeBtn.Location = new
Point(Width - btnWidth - 0, 0); maximizeBtn.Location = new
Point(Width - btnWidth * 2 - 0, 0); minimizeBtn.Location = new
Point(Width - btnWidth * 3 - 0, 0); };

        mainTabControl = new TabControl { Dock =
DockStyle.Fill, Font = new DrawingFont("Segoe UI", 11,
System.Drawing.FontStyle.Bold) };
        var searchTab = new TabPage("🔍 Поиск");
InitializeSearchTab(searchTab);
        var recommendationsTab = new
TabPage("📌 Рекомендации");
InitializeRecommendationsTab(recommendationsTab);
        var favoritesTab = new TabPage("★ Избранное");
InitializeFavoritesTab(favoritesTab);
        var profileTab = new TabPage("👤 Профиль");
InitializeProfileTab(profileTab);
        avatarPictureBox.Cursor = Cursors.Hand;
        mainTabControl.TabPages.AddRange(new[] { searchTab,
recommendationsTab, favoritesTab, profileTab });
        mainTabControl.SelectedIndexChanged +=
MainTabControl_SelectedIndexChanged;
        Controls.Add(mainTabControl);
        Controls.Add(titleBar);
    }

    protected override void WndProc(ref Message m)
    {
        const int WM_NCHITTEST = 0x84;
        const int HTLEFT = 10, HTRIGHT = 11, HTTOP = 12,
HTTOPLEFT = 13, HTTOPRIGHT = 14, HTBOTTOM = 15, HTBOTTOMLEFT =
16, HTBOTTOMRIGHT = 17;
        if (m.Msg == WM_NCHITTEST)

```

```

        {
            base.WndProc(ref m);
            Point pt = PointToClient(new
Point(m.LParam.ToInt32() & 0xffff, m.LParam.ToInt32() >> 16));
            Size clientSize = ClientSize;
            int borderWidth = 5;
            if (pt.X <= borderWidth && pt.Y <= borderWidth)
m.Result = (IntPtr)HTTOPLEFT;
            else if (pt.X >= clientSize.Width - borderWidth
&& pt.Y <= borderWidth) m.Result = (IntPtr)HTTOPRIGHT;
            else if (pt.X <= borderWidth && pt.Y >=
clientSize.Height - borderWidth) m.Result = (IntPtr)HTBOTTOMLEFT;
            else if (pt.X >= clientSize.Width - borderWidth
&& pt.Y >= clientSize.Height - borderWidth) m.Result =
(IntPtr)HTBOTTOMRIGHT;
            else if (pt.Y <= borderWidth) m.Result =
(IntPtr)HTTOP;
            else if (pt.Y >= clientSize.Height - borderWidth)
m.Result = (IntPtr)HTBOTTOM;
            else if (pt.X <= borderWidth) m.Result =
(IntPtr)HTLEFT;
            else if (pt.X >= clientSize.Width - borderWidth)
m.Result = (IntPtr)HTRIGHT;
            else return;
        }
        else base.WndProc(ref m);
    }

    private void InitializeRecommendationsTab(TabPage tab)
    {
        tab.Padding = new Padding(15);
        var titlePanel = new WinFormsPanel { Dock =
DockStyle.Top, Height = 80, Padding = new Padding(0) };
        var titleLabel = new WinFormsLabel { Text =
"🔍 РЕКОМЕНДАЦИИ ДЛЯ ВАС", Font = new DrawingFont("Segoe UI", 18,
System.Drawing.FontStyle.Bold), Location = new Point(0, 10),
AutoSize = true };
        var descriptionLabel = new WinFormsLabel { Text =
"На основе ваших предпочтений и вкусов похожих читателей", Font = new
DrawingFont("Segoe UI", 10), ForeColor = DrawingColor.Gray,
Location = new Point(0, 45), AutoSize = true };
        var refreshButton = new ModernButton { Text =
"🔄 ОБНОВИТЬ", Size = new Size(140, 35), Location = new Point(370,
10), NormalColor = DrawingColor.FromArgb(0, 120, 215),
BorderRadius = 5, Anchor = AnchorStyles.Top |
AnchorStyles.Left };
        refreshButton.Click += async (s, e) =>
{ recommendationsLoaded = false; await LoadRecommendations();
recommendationsLoaded = true; };
        titlePanel.Controls.Add(titleLabel);
        titlePanel.Controls.Add(descriptionLabel);
    }

```

```

        titlePanel.Controls.Add(refreshButton);
        recommendationsPanel = new FlowLayoutPanel { Dock =
DockStyle.Fill, AutoScroll = true, Padding = new Padding(20),
BackColor = DrawingColor.Transparent };
        tab.Controls.Add(recommendationsPanel);
        tab.Controls.Add(titlePanel);
    }

    private async Task LoadRecommendations()
    {
        try
        {
            recommendationsPanel.Controls.Clear();
            var loadingLabel = new WinFormsLabel { Text =
"Анализируем ваши предпочтения...", Font = new DrawingFont("Segoe
UI", 12), ForeColor = currentTheme == "Темная" ?
DrawingColor.White : DrawingColor.Black, AutoSize = true,
Location = new Point(20, 20) };
            recommendationsPanel.Controls.Add(loadingLabel);
            await Task.Delay(300);
            var recommendations =
dbManager.GetRecommendations(currentUsername, 20);
            if (contentFilterEnabled && recommendations !=
null) recommendations = FilterBooks(recommendations).ToList();
            if (recommendations != null &&
recommendations.Count > 0) recommendations =
recommendations.Take(10).ToList();
            else recommendations = new List<Book>();
            recommendationsPanel.Controls.Clear();
            if (recommendations.Count == 0)
            {
                var noRecsLabel = new WinFormsLabel { Text =
dbManager.CountUserFavorites(currentUsername) < 3 ? "📖 Добавьте
больше книг в избранное (минимум 3), чтобы получить
рекомендации!" : (contentFilterEnabled ? "📖 На основе ваших
предпочтений не найдено подходящих книг.\n(Возможно, некоторые
книги были скрыты фильтром контента)" : "📖 На основе ваших
предпочтений не найдено подходящих книг.\nПопробуйте добавить
другие книги в избранное!"), Font = new DrawingFont("Segoe UI",
12), ForeColor = DrawingColor.Gray, AutoSize = false, Size = new
Size(500, 80), Location = new Point(20, 20), TextAlign =
ContentAlignment.MiddleLeft };
                recommendationsPanel.Controls.Add(noRecsLabel);
                return;
            }
            recommendationsPanel.Padding = new Padding(15);
            foreach (var book in recommendations)
            {
                try
                {
                    DrawingImage cover = null;

```

```

        if (book.CoverId.HasValue)
        {
            var cachedCover =
dbManager.GetCachedCover(book.CoverId.ToString());
            if (cachedCover != null) { using var
ms = new MemoryStream(cachedCover); cover =
DrawingImage.FromStream(ms); }
            else
                try { var bytes = await
httpClient.GetByteArrayAsync($"{COVER_IMAGE_URL}/id/{book.CoverId
}-M.jpg"); dbManager.CacheCover(book.CoverId.ToString(), bytes);
using var ms = new MemoryStream(bytes); cover =
DrawingImage.FromStream(ms); } catch { }
        }
        bool isInFavorites =
dbManager.IsBookInFavorites(currentUsername, book.Key ?? "");
        var card = new BookCard(book, cover,
isInFavorites);
        card.BookCardClicked += (s, clickedBook)
=> OpenBookInBrowser(clickedBook);
        card.AddToFavoritesClicked += async (s,
b) => {
            if
(!dbManager.IsBookInFavorites(currentUsername, b.Key ?? ""))
            {
                if (contentFilterEnabled
&& !IsBookSafe(b))
                    if
(MessageBox.Show("Эта книга может содержать контент 18+. Вы уверены,
что хотите добавить её в избранное?", "Предупреждение",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning) !=
DialogResult.Yes) return;

                dbManager.SaveFavorite(currentUsername, b.Title, b.AuthorNames ??
"Неизвестен", b.FirstPublishYear?.ToString() ?? "?",
DateTime.Now.ToString("dd.MM.yyyy HH:mm"), b.Key ?? "",
b.CoverId?.ToString() ?? "", b.Rating, b.Description);
                RefreshFavoritesCount();
                if (s is BookCard bookCard)
bookCard.UpdateFavoriteStatus(true);
                MessageBox.Show($"Книга
'{b.Title}' добавлена в избранное!", "Успех", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
        };
        recommendationsPanel.Controls.Add(card);
    }
    catch (Exception ex)
    {
        Debug.WriteLine($"Ошибка при создании карточки книги:
{ex.Message}");
    }
}

```

```

        catch (Exception ex)
        {
            recommendationsPanel.Controls.Clear();
            var errorLabel = new WinFormsLabel { Text =
$"Ошибка при загрузке рекомендаций: {ex.Message}", Font = new
DrawingFont("Segoe UI", 12), ForeColor = DrawingColor.Red,
AutoSize = false, Size = new Size(500, 60), Location = new
Point(20, 20) };
            recommendationsPanel.Controls.Add(errorLabel);
        }
    }

    private void LoadAvatar()
    {
        try
        {
            byte[] avatarData =
dbManager.LoadAvatar(currentUsername);
            if (avatarData != null && avatarData.Length > 0)
                using (var ms = new MemoryStream(avatarData))
                {
                    var originalImage =
DrawingImage.FromStream(ms);
                    avatarPictureBox.Image =
ResizeImage(originalImage, 128, 128);
                    originalImage.Dispose();
                }
            else avatarPictureBox.Image =
CreateDefaultAvatar();
        }
        catch { avatarPictureBox.Image =
CreateDefaultAvatar(); }
    }

    private DrawingImage CreateDefaultAvatar()
    {
        Bitmap bmp = new Bitmap(80, 80);
        using (Graphics g = Graphics.FromImage(bmp))
        {
            g.Clear(DrawingColor.FromArgb(0, 120, 215));
            using (var font = new DrawingFont("Segoe UI", 32,
System.Drawing.FontStyle.Bold))
                using (var brush = new
SolidBrush(DrawingColor.White))
                {
                    var sf = new StringFormat { Alignment =
StringAlignment.Center, LineAlignment = StringAlignment.Center };
                    string initials = currentUsername.Length >
0 ? currentUsername[0].ToString().ToUpper() : "?";
                    g.DrawString(initials, font, brush, new
System.Drawing.Rectangle(0, 0, 80, 80), sf);
                }
        }
    }

```

```

        }
        return bmp;
    }

    private void ChangeAvatar_Click(object sender, EventArgs
e)
    {
        using (OpenFileDialog openFileDialog = new
OpenFileDialog() { Filter = "Image
Files|*.jpg;*.jpeg;*.png;*.bmp;*.gif", Title =
"Выберите изображение для аватарки" })
            if (openFileDialog.ShowDialog() ==
DialogResult.OK)
                try
                {
                    using (var originalImage =
DrawingImage.FromFile(openFileDialog.FileName))
                    {
                        byte[] originalImageBytes;
                        using (var ms = new MemoryStream())
                        { originalImage.Save(ms, System.Drawing.Imaging.ImageFormat.Png);
originalImageBytes = ms.ToArray(); }
                        dbManager.SaveAvatar(currentUsername,
originalImageBytes);
                        avatarPictureBox.Image =
ResizeImage(originalImage, 128, 128);

                        MessageBox.Show("Аватарка успешно обновлена!", "Успех",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
                catch (Exception ex)
                { MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error); }
            }

    private DrawingImage ResizeImage(DrawingImage image, int
width, int height)
    {
        var destImage = new Bitmap(width, height);
        destImage.SetResolution(image.HorizontalResolution,
image.VerticalResolution);
        using (var graphics = Graphics.FromImage(destImage))
        {
            graphics.CompositingQuality =
CompositingQuality.HighQuality;
            graphics.InterpolationMode =
InterpolationMode.HighQualityBicubic;
            graphics.SmoothingMode =
SmoothingMode.HighQuality;
            graphics.PixelOffsetMode =
PixelOffsetMode.HighQuality;

```

```

        graphics.DrawImage(image, new
System.Drawing.Rectangle(0, 0, width, height), 0, 0, image.Width,
image.Height, GraphicsUnit.Pixel);
    }
    return destImage;
}

private void OpenBookInBrowser(Book book)
{
    if (book == null) return;
    if (contentFilterEnabled && !IsBookSafe(book))
        if
(MessageBox.Show("Эта книга может содержать контент 18+. Вы уверены,
что хотите открыть её в браузере?", "Предупреждение",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning) !=
DialogResult.Yes) return;
    try
    {
        string url = !string.IsNullOrEmpty(book.Url) ?
book.Url : $"{{OPEN_LIBRARY_BOOK_URL}}{{book.Key}}";
        if (!string.IsNullOrEmpty(url))
        {
            Process.Start(new ProcessStartInfo(url)
{ UseShellExecute = true });
            dbManager.SaveViewHistory(currentUsername,
book.Key ?? "");
        }
        else
        {
            MessageBox.Show("Не удалось открыть страницу книги.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка при открытии браузера: {ex.Message}",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void InitializeSearchTab(TabPage tab)
    {
        tab.Padding = new Padding(15);
        var topPanel = new WinFormsPanel { Dock =
DockStyle.Top, Height = 50 };
        welcomeLabel = new WinFormsLabel { Text =
$"Добро пожаловать, {currentUsername}!", Font = new
DrawingFont("Segoe UI", 14, System.Drawing.FontStyle.Bold), Dock =
DockStyle.Left, AutoSize = true, Padding = new Padding(10, 10,
0, 0) };
        topPanel.Controls.Add(welcomeLabel);
        var searchPanel = new WinFormsPanel { Dock =
DockStyle.Top, Height = 70, Padding = new Padding(0, 10, 0,
10) };
        searchBox = new WinFormsTextBox { Location = new
Point(10, 12), Size = new Size(500, 45), Font = new

```

```

DrawingFont("Segoe UI", 13), Text = "программирование",
BorderStyle = BorderStyle.FixedSingle };

        searchButton = new ModernButton { Text = "🔍 ПОИСК",
Location = new Point(520, 12), Size = new Size(140, 45),
NormalColor = DrawingColor.FromArgb(0, 120, 215) };
        searchButton.Click += SearchButton_Click;
        addToFavoritesButton = new ModernButton { Text =
"★ В ИЗБРАННОЕ", Location = new Point(670, 12), Size = new
Size(160, 45), NormalColor = DrawingColor.Gold, HoverColor =
DrawingColor.Orange, ForeColor = DrawingColor.Black, Enabled =
false };

        addToFavoritesButton.Click +=
AddToFavoritesButton_Click;
        searchPanel.Controls.AddRange(new Control[]
{ searchBox, searchButton, addToFavoritesButton });
        var mainSplit = new SplitContainer { Dock =
DockStyle.Fill, Orientation =
System.Windows.Forms.Orientation.Vertical, SplitterDistance =
600, SplitterWidth = 8 };
        searchResultsListView = new ListView { Dock =
DockStyle.Fill, View = View.Details, FullRowSelect = true,
GridLines = true, MultiSelect = false, Font = new
DrawingFont("Segoe UI", 10) };
        searchResultsListView.Columns.Add("Название", 350);
        searchResultsListView.Columns.Add("Автор", 250);
        searchResultsListView.Columns.Add("Год", 80);
        searchResultsListView.Columns.Add("Рейтинг", 80);
        searchResultsListView.SelectedIndexChanged +=
SearchResultsListView_SelectedIndexChanged;
        searchResultsListView.DoubleClick +=
SearchResultsListView_DoubleClick;
        var rightPanel = new WinFormsPanel { Dock =
DockStyle.Fill, Padding = new Padding(10, 0, 0, 0) };
        var rightSplit = new SplitContainer { Dock =
DockStyle.Fill, Orientation =
System.Windows.Forms.Orientation.Horizontal, SplitterDistance =
300, SplitterWidth = 8 };
        var detailsPanel = new WinFormsPanel { Dock =
DockStyle.Fill };
        bookDetailsTextBox = new RichTextBox { Dock =
DockStyle.Fill, ReadOnly = true, Font = new DrawingFont("Segoe
UI", 11), BorderStyle = BorderStyle.None, Padding = new
Padding(12) };
        detailsPanel.Controls.Add(bookDetailsTextBox);
        bookDetailsTextBox.LinkClicked +=
BookDetailsTextBox_LinkClicked;
        bookDetailsTextBox.DetectUrls = true;
        var coverPanel = new WinFormsPanel { Dock =
DockStyle.Fill, Padding = new Padding(12) };
        bookCoverPicture = new WinFormsPictureBox { Dock =
DockStyle.Fill, SizeMode = PictureBoxSizeMode.Zoom };

```

```

        coverPanel.Controls.Add(bookCoverPicture);
        rightSplit.Panel1.Controls.Add(detailsPanel);
        rightSplit.Panel2.Controls.Add(coverPanel);
        rightPanel.Controls.Add(rightSplit);
        mainSplit.Panel1.Controls.Add(searchResultsListView);
        mainSplit.Panel2.Controls.Add(rightPanel);
        var statusPanel = new WinFormsPanel { Dock =
DockStyle.Bottom, Height = 40 };
        statusLabel = new WinFormsLabel { Text =
"Готовкпоиску...", Dock = DockStyle.Left, Width = 700, Padding =
new Padding(15, 10, 0, 0), Font = new DrawingFont("Segoe UI",
10) };

        progressBar = new WinFormsProgressBar { Dock =
DockStyle.Right, Width = 200, Style = ProgressBarStyle.Marquee,
Visible = false, Margin = new Padding(0, 5, 15, 5) };
        statusPanel.Controls.Add(progressBar);
        statusPanel.Controls.Add(statusLabel);
        tab.Controls.Add(mainSplit);
        tab.Controls.Add(statusPanel);
        tab.Controls.Add(searchPanel);
        tab.Controls.Add(topPanel);
        searchBox.KeyPress += (s, e) => { if (e.KeyChar ==
(char)Keys.Enter) { SearchButton_Click(s, e); e.Handled =
true; } };
    }

    private void InitializeFavoritesTab(TabPage tab)
    {
        tab.Padding = new Padding(15);
        var titleLabel = new WinFormsLabel { Text =
$"★ИЗБРАННОЕ {currentUsername}", Font = new DrawingFont("Segoe
UI", 18, System.Drawing.FontStyle.Bold), Dock = DockStyle.Top,
Height = 60, TextAlign = ContentAlignment.MiddleCenter };
        var controlPanel = new WinFormsPanel { Dock =
DockStyle.Top, Height = 60, Padding = new Padding(10, 10, 0,
0) };

        removeFromFavoritesButton = new ModernButton { Text =
"❌УДАЛИТЬВЫБРАННОЕ", Location = new Point(10, 8), Size = new
Size(220, 45), NormalColor = DrawingColor.FromArgb(220, 80, 80),
Enabled = false };
        removeFromFavoritesButton.Click +=
RemoveFromFavoritesButton_Click;
        clearFavoritesButton = new ModernButton { Text =
"△ОЧИСТИТЬВСЁ", Location = new Point(240, 8), Size = new
Size(160, 45), NormalColor = DrawingColor.FromArgb(255, 140,
0) };
        clearFavoritesButton.Click +=
ClearFavoritesButton_Click;
        exportPdfButton = new ModernButton { Text =
"📄ЭКСПОРТВ PDF", Location = new Point(410, 8), Size = new

```

```

Size(180, 45), NormalColor = DrawingColor.FromArgb(46, 204,
113) };

        exportPdfButton.Click += ExportToPdfButton_Click;
        controlPanel.Controls.AddRange(new Control[]
{ removeFromFavoritesButton, clearFavoritesButton,
exportPdfButton });
        favoritesListView = new ListView { Dock =
DockStyle.Fill, View = View.Details, FullRowSelect = true,
GridLines = true, Font = new DrawingFont("Segoe UI", 10) };
        favoritesListView.Columns.Add("Название", 400);
        favoritesListView.Columns.Add("Автор", 250);
        favoritesListView.Columns.Add("Год", 100);
        favoritesListView.Columns.Add("Датадобавления", 180);
        favoritesListView.SelectedIndexChanged +=
FavoritesListView_SelectedIndexChanged;
        favoritesListView.DoubleClick +=
FavoritesListView_DoubleClick;
        tab.Controls.Add(favoritesListView);
        tab.Controls.Add(controlPanel);
        tab.Controls.Add(titleLabel);
    }

    private void InitializeProfileTab(TabPage tab)
    {
        tab.Padding = new Padding(20);
        tab.AutoScroll = true;

        var mainPanel = new WinFormsPanel { Dock =
DockStyle.Fill, AutoScroll = true };

        var titleLabel = new WinFormsLabel
        {
            Text = "👤 ПРОФИЛЬ И СТАТИСТИКА",
            Font = new DrawingFont("Segoe UI", 20,
System.Drawing.FontStyle.Bold),
            Dock = DockStyle.Top,
            Height = 50,
            TextAlign = ContentAlignment.MiddleCenter
        };

        // Верхняя панель с информацией и настройками
        var topPanel = new TableLayoutPanel
        {
            Dock = DockStyle.Top,
            Height = 280,
            ColumnCount = 2,
            RowCount = 1,
            Padding = new Padding(0, 0, 0, 5)
        };
        topPanel.ColumnStyles.Add(new
ColumnStyle(SizeType.Percent, 50));

```

```

        topPanel.ColumnStyles.Add(new
ColumnStyle(SizeType.Percent, 50));

// ===== ЛЕВАЯ ПАНЕЛЬ - ИНФОРМАЦИЯ О ПОЛЬЗОВАТЕЛЕ =====
var userInfoPanel = new WinFormsPanel
{
    Dock = DockStyle.Fill,
    Padding = new Padding(10)
};

var userIcon = new WinFormsLabel
{
    Text = " ",
    Font = new DrawingFont("Segoe UI", 64),
    Location = new Point(15, 15),
    Size = new Size(90, 90)
};

var usernameLabel = new WinFormsLabel
{
    Text = currentUsername,
    Font = new DrawingFont("Segoe UI", 16,
System.Drawing.FontStyle.Bold),
    Location = new Point(130, 20),
    AutoSize = true
};

currentThemeLabel = new WinFormsLabel
{
    Text = $"Тема: {currentTheme}",
    Font = new DrawingFont("Segoe UI", 11),
    Location = new Point(130, 55),
    AutoSize = true
};

avatarPictureBox = new PictureBox
{
    Size = new Size(128, 128),
    Location = new Point(0, 15),
    SizeMode = PictureBoxSizeMode.Zoom,
    BackColor = DrawingColor.FromArgb(220, 220, 220),
    BorderStyle = BorderStyle.None,
    Cursor = Cursors.Hand
};

avatarPictureBox.Click += (s, e) =>
ShowAvatarFullScreen();

changeAvatarButton = new Button
{
    Text = "📷 Сменить фото",
    Size = new Size(128, 30),

```

```

        Location = new Point(0, 148),
        BackColor = DrawingColor.FromArgb(0, 120, 215),
        ForeColor = DrawingColor.White,
        FlatStyle = FlatStyle.Flat,
        Font = new DrawingFont("Segoe UI", 9,
System.Drawing.FontStyle.Bold),
        Cursor = Cursors.Hand
    };
changeAvatarButton.FlatAppearance.BorderSize = 0;
changeAvatarButton.Click += ChangeAvatar_Click;

userInfoPanel.Controls.Add(avatarPictureBox);
userInfoPanel.Controls.Add(changeAvatarButton);

userAgeLabel = new WinFormsLabel
{
    Text = $"Возраст: {userAge} лет {(userAge < 18 ?
" (18) Ограниченный доступ" : "(✓ Полный доступ)) }",
    Font = new DrawingFont("Segoe UI", 11),
    ForeColor = userAge < 18 ? DrawingColor.Orange :
DrawingColor.LightGreen,
    Location = new Point(130, 85),
    AutoSize = true
};

filterStatusLabel = new WinFormsLabel
{
    Text = $"Фильтр контента: {(contentFilterEnabled ?
"✓ Включен" : "✗ Отключен") }",
    Font = new DrawingFont("Segoe UI", 11),
    ForeColor = contentFilterEnabled ?
DrawingColor.LightGreen : DrawingColor.Orange,
    Location = new Point(130, 110),
    AutoSize = true
};

favoritesCountLabel = new WinFormsLabel
{
    Text = $"📖 Книг в избранном:
{dbManager.CountUserFavorites(currentUsername)}",
    Font = new DrawingFont("Segoe UI", 11),
    Location = new Point(130, 140),
    AutoSize = true
};

var emailLabel = new WinFormsLabel
{
    Text = $"✉ Email:
{dbManager.GetUserEmail(currentUsername)}",
    Font = new DrawingFont("Segoe UI", 11),
    Location = new Point(130, 170),

```

```

        AutoSize = true
    };

    var changeEmailButton = new ModernButton
    {
        Text = "✉ Изменить Email",
        Size = new Size(160, 30),
        Location = new Point(130, 200),
        NormalColor = DrawingColor.FromArgb(0, 120, 215),
        Font = new DrawingFont("Segoe UI", 9)
    };
    changeEmailButton.Click += (s, e) => {
        string newEmail =
Microsoft.VisualBasic.Interaction.InputBox(
    "Введите новый Email:",
        "Изменение Email",
    dbManager.GetUserEmail(currentUsername));

        if (!string.IsNullOrEmpty(newEmail) &&
newEmail.Contains("@"))
        {
            dbManager.UpdateUserEmail(currentUsername,
newEmail);

            UpdateProfileEmail();
            MessageBox.Show("Email обновлён!", "Успех",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        else if (!string.IsNullOrEmpty(newEmail))
        {
            MessageBox.Show("Введите корректный Email!",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    };

    userInfoPanel.Controls.AddRange(new Control[]
    {
        userIcon, usernameLabel, currentThemeLabel, userAgeLabel,
        filterStatusLabel, favoritesCountLabel, emailLabel,
changeEmailButton
    });

    // ===== ПРАВАЯ ПАНЕЛЬ - НАСТРОЙКИ =====
    var settingsPanel = new WinFormsPanel
    {
        Dock = DockStyle.Fill,
        Padding = new Padding(10),
        AutoScroll = true
    };

    var themeTitle = new WinFormsLabel
    {

```

```

        Text = "☹️Тема оформления",
        Font = new DrawingFont("Segoe UI", 12,
System.Drawing.FontStyle.Bold),
        Location = new Point(10, 10),
        AutoSize = true
    };

    var btnLight = new ModernButton
    {
        Text = "☀️Светлая",
        Size = new Size(110, 40),
        Location = new Point(10, 45),
        NormalColor = DrawingColor.FromArgb(200, 200,
200),
        ForeColor = DrawingColor.Black
    };
    btnLight.Click += (s, e) => ApplyNewTheme("Светлая");

    var btnDark = new ModernButton
    {
        Text = "☾Темная",
        Size = new Size(110, 40),
        Location = new Point(130, 45),
        NormalColor = DrawingColor.FromArgb(60, 60, 60)
    };
    btnDark.Click += (s, e) => ApplyNewTheme("Темная");

    var btnForest = new ModernButton
    {
        Text = "🌲Лесная",
        Size = new Size(110, 40),
        Location = new Point(250, 45),
        NormalColor = DrawingColor.FromArgb(76, 175, 80)
    };
    btnForest.Click += (s, e) => ApplyNewTheme("Лесная");

    settingsPanel.Controls.Add(themeTitle);
    settingsPanel.Controls.Add(btnLight);
    settingsPanel.Controls.Add(btnDark);
    settingsPanel.Controls.Add(btnForest);

    var filterTitle = new WinFormsLabel
    {
        Text = "Ⓜ️Настройки контента",
        Font = new DrawingFont("Segoe UI", 12,
System.Drawing.FontStyle.Bold),
        Location = new Point(10, 100),
        AutoSize = true
    };

    var toggleFilterBtn = new ModernButton

```

```

        {
            Text = contentFilterEnabled ? "18 Отключить фильтр
18+" : "18 Включить фильтр 18+",
            Size = new Size(200, 40),
            Location = new Point(10, 130),
            NormalColor = contentFilterEnabled ?
DrawingColor.FromArgb(231, 76, 60) : DrawingColor.FromArgb(46,
204, 113),
            ForeColor = DrawingColor.White
        };

        if (userAge < 18)
        {
            toggleFilterBtn.Enabled = false;
            toggleFilterBtn.Text = "18 Фильтр 18+
(обязателен)";
            toggleFilterBtn.NormalColor =
DrawingColor.FromArgb(100, 100, 100);
        }
        else
        {
            toggleFilterBtn.Click += ToggleContentFilter;
        }

        var filterInfoLabel = new WinFormsLabel
    {
        Text = "Фильтр скрывает книги с маркировкой 18+",
        Font = new DrawingFont("Segoe UI", 8),
        ForeColor = DrawingColor.Gray,
        Location = new Point(10, 175),
        Size = new Size(300, 30),
        AutoSize = true
    };

        settingsPanel.Controls.Add(filterTitle);
        settingsPanel.Controls.Add(toggleFilterBtn);
        settingsPanel.Controls.Add(filterInfoLabel);

        topPanel.Controls.Add(userInfoPanel, 0, 0);
        topPanel.Controls.Add(settingsPanel, 1, 0);

        // ===== ГРАФИК СТАТИСТИКИ (поднять выше)
        =====
        var statsPanel = new WinFormsPanel
        {
            Dock = DockStyle.Top,
            Height = 320,
            Padding = new Padding(10, 5, 10, 5)
        };

        var statsTitle = new WinFormsLabel

```

```

{
    Text = "📊 СТАТИСТИКА ДОБАВЛЕНИЯ КНИГ",
    Font = new DrawingFont("Segoe UI", 14,
        System.Drawing.FontStyle.Bold),
    Dock = DockStyle.Top,
    Height = 30
};

var plotView = new FormsPlot { Dock =
DockStyle.Fill };
plotView.Name = "statsPlot";

plotView.UserInputProcessor.IsEnabled = false;
RefreshStatsPlot(plotView);

statsPanel.Controls.Add(plotView);
statsPanel.Controls.Add(statsTitle);

// ===== КНОПКА ВЫХОДА =====
var actionsPanel = new WinFormsPanel
{
    Dock = DockStyle.Top,
    Height = 60,
    Padding = new Padding(10, 5, 10, 5)
};

logoutButton = new ModernButton
{
    Text = "🚪 ВЫЙТИ ИЗ АККАУНТА",
    Size = new Size(220, 45),
    Location = new Point(10, 5),
    NormalColor = DrawingColor.FromArgb(200, 60, 60)
};
logoutButton.Click += LogoutButton_Click;
actionsPanel.Controls.Add(logoutButton);

// ===== СБОРКА =====
mainPanel.Controls.Add(actionsPanel);
mainPanel.Controls.Add(statsPanel);
mainPanel.Controls.Add(topPanel);
mainPanel.Controls.Add(titleLabel);

tab.Controls.Add(mainPanel);

LoadAvatar();
}
private void ToggleContentFilter(object sender, EventArgs
e)
{

```

```

        if (userAge < 18) { MessageBox.Show("Вам меньше 18
лет. Фильтр контента не может быть отключен.", "Ограничение доступа",
MessageBoxButtons.OK, MessageBoxIcon.Warning); return; }
        contentFilterEnabled = !contentFilterEnabled;
        dbManager.UpdateContentFilter(currentUsername,
contentFilterEnabled);
        if (sender is ModernButton btn) { btn.Text =
contentFilterEnabled ? "Ⓜ Отключить фильтр 18+" :
"Ⓜ Включить фильтр 18+"; btn.NormalColor = contentFilterEnabled ?
DrawingColor.FromArgb(231, 76, 60) : DrawingColor.FromArgb(46,
204, 113); }
        filterStatusLabel.Text = $"Фильтр контента:
{(contentFilterEnabled ? "✓ Включен" : "✗ Отключен")}";
        filterStatusLabel.ForeColor = contentFilterEnabled ?
DrawingColor.Green : DrawingColor.Orange;
        MessageBox.Show($"Фильтр контента
{(contentFilterEnabled ? "включен" : "отключен")}", "Настройки",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        if (searchResultsListView.Items.Count > 0)
SearchButton_Click(sender, e);
    }

    private IEnumerable<Book> FilterBooks(IEnumerable<Book>
books)
    {
        foreach (var book in books) if (IsBookSafe(book))
yield return book;
    }

    private bool IsBookSafe(Book book)
    {
        if (userAge < 18)
        {
            if (!string.IsNullOrEmpty(book.Title)) foreach
(var keyword in AdultKeywords) if
(book.Title.ToLower().Contains(keyword.ToLower())) return false;
            if (!string.IsNullOrEmpty(book.AuthorNames))
foreach (var keyword in AdultKeywords) if
(book.AuthorNames.ToLower().Contains(keyword.ToLower())) return
false;
            if (!string.IsNullOrEmpty(book.Description))
foreach (var keyword in AdultKeywords) if
(book.Description.ToLower().Contains(keyword.ToLower())) return
false;
            if (book.Subjects != null) foreach (var subject
in book.Subjects) if (!string.IsNullOrEmpty(subject)) foreach
(var keyword in AdultKeywords) if
(subject.ToLower().Contains(keyword.ToLower())) return false;
            return true;
        }
        if (!contentFilterEnabled) return true;
    }

```

```

        if (!string.IsNullOrEmpty(book.Title)) foreach (var
keyword in AdultKeywords) if
(book.Title.ToLower().Contains(keyword.ToLower())) return false;
        if (!string.IsNullOrEmpty(book.AuthorNames)) foreach
(var keyword in AdultKeywords) if
(book.AuthorNames.ToLower().Contains(keyword.ToLower())) return
false;
        if (!string.IsNullOrEmpty(book.Description)) foreach
(var keyword in AdultKeywords) if
(book.Description.ToLower().Contains(keyword.ToLower())) return
false;
        if (book.Subjects != null) foreach (var subject in
book.Subjects) if (!string.IsNullOrEmpty(subject)) foreach (var
keyword in AdultKeywords) if
(subject.ToLower().Contains(keyword.ToLower())) return false;
        return true;
    }

    private void ExportToPdfButton_Click(object sender,
EventArgs e)
    {
        var favorites =
dbManager.LoadFavorites(currentUsername);
        if (favorites.Count == 0)
        {
            MessageBox.Show("Нет книг для экспорта!",
"Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);
            return;
        }

        var saveDialog = new SaveFileDialog
        {
            Filter = "PDF Files|*.pdf",
            FileName =
$"Избранное_{currentUsername}_{DateTime.Now:yyyyMMdd}.pdf"
        };

        if (saveDialog.ShowDialog() == DialogResult.OK)
        {
            try
            {
                // Путь к шрифту Arial в Windows
                (гарантированно поддерживает русский язык)
                string fontPath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.
Fonts), "arial.ttf");

                // Если Arial не найден (например, на Linux), используем
                // встроенный базовый шрифт с поддержкой Unicode
                BaseFont baseFont;

                if (File.Exists(fontPath))
                {

```

```

        baseFont = BaseFont.CreateFont(fontPath,
BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
    }
    else
    {
        // Fallback на настроенный шрифт Helvetica
споддержкой Unicode
        baseFont =
BaseFont.CreateFont(BaseFont.HELVETICA, BaseFont.IDENTITY_H,
BaseFont.EMBEDDED);
    }

    using (var fs = new
FileStream(saveDialog.FileName, FileMode.Create))
    {
        var document = new
iTextSharp.text.Document();
        PdfWriter.GetInstance(document, fs);
        document.Open();

        // Создаем шрифты для разных элементов
        var titleFont = new
iTextSharp.text.Font(baseFont, 18, iTextSharp.text.Font.BOLD);
        var headerFont = new
iTextSharp.text.Font(baseFont, 12, iTextSharp.text.Font.BOLD);
        var cellFont = new
iTextSharp.text.Font(baseFont, 10, iTextSharp.text.Font.NORMAL);
        var infoFont = new
iTextSharp.text.Font(baseFont, 8, iTextSharp.text.Font.ITALIC);

        // Заголовок
        var title = new
Paragraph($"Список избранных книг пользователя {currentUsername}",
titleFont);

        title.Alignment = Element.ALIGN_CENTER;
        title.SpacingAfter = 20;
        document.Add(title);

        // Таблица
        var table = new PdfPTable(4);
        table.WidthPercentage = 100;
        table.SetWidths(new float[] { 40, 30, 15,
15 });

        // Заголовки таблицы
        table.AddCell(new PdfPCell(new
Phrase("Название", headerFont)));
        table.AddCell(new PdfPCell(new
Phrase("Автор", headerFont)));
        table.AddCell(new PdfPCell(new
Phrase("Год", headerFont)));

```

```

        table.AddCell(new PdfPCell(new
Phrase("Добавлено", headerFont)));

        // Данные
        foreach (var book in favorites)
        {
            table.AddCell(new PdfPCell(new
Phrase(book.Title ?? "", cellFont)));
            table.AddCell(new PdfPCell(new
Phrase(book.Author ?? "", cellFont)));
            table.AddCell(new PdfPCell(new
Phrase(book.Year ?? "", cellFont)));
            table.AddCell(new PdfPCell(new
Phrase(book.AddedDate ?? "", cellFont)));
        }

        document.Add(table);

        // Информация о дате экспорта
        var info = new
Paragraph($"\\n\\nЭкспортировано: {DateTime.Now:dd.MM.yyyy HH:mm}",
infoFont);
        info.Alignment = Element.ALIGN_RIGHT;
        document.Add(info);

        document.Close();
    }

    MessageBox.Show($"PDF
успешно создан!\\n{saveDialog.FileName}", "Успех",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    Process.Start(new
ProcessStartInfo(saveDialog.FileName) { UseShellExecute =
true });
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка при создании PDF:
{ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}

private async void SearchButton_Click(object sender,
EventArgs e)
{
    string query = searchBox.Text.Trim();
    if (string.IsNullOrEmpty(query)) return;
    SetSearchState(true);
    searchResultsListView.Items.Clear();
    bookDetailsTextBox.Clear();
}

```

```

bookCoverPicture.Image = null;
addToFavoritesButton.Enabled = false;
currentBooks.Clear();
try
{
    statusLabel.Text = $"Поиск «{query}»...";
    var books = await SearchOpenLibrary(query, 50);
    if (contentFilterEnabled)
    {
        var filteredBooks = new List<Book>();
        int blockedCount = 0;
        foreach (var book in books) if
(IsBookSafe(book)) filteredBooks.Add(book); else blockedCount++;
        currentBooks =
filteredBooks.Take(20).ToList();
        statusLabel.Text = blockedCount > 0 ?
$"Найдено {currentBooks.Count} книг (скрыто {blockedCount} книг
18+)" : $"Найдено {currentBooks.Count} книг (фильмативен)";
    }
    else { currentBooks = books.Take(20).ToList();
statusLabel.Text = $"Найдено {currentBooks.Count} книг
(фильмативен)"; }
    if (currentBooks.Count > 0)
    {
        foreach (var book in currentBooks)
        {
            var item = new ListViewItem(new[]
{ book.Title.Length > 55 ? book.Title.Substring(0, 52) + "..." :
book.Title, (book.AuthorNames?.Length > 32 ?
book.AuthorNames.Substring(0, 29) + "..." : book.AuthorNames) ??
"Неизвестен", book.FirstPublishYear?.ToString() ?? "-",
book.Rating.HasValue ? new string('★', (int)book.Rating.Value) :
"☆☆☆☆☆" }) { Tag = book };
            searchResultsListView.Items.Add(item);
        }
        if (searchResultsListView.Items.Count > 0)
searchResultsListView.Items[0].Selected = true;
    }

    else statusLabel.Text = contentFilterEnabled ?
"Ничего не найдено (все результаты скрыты фильмом 18+)" :
"Ничего не найдено";
}
catch (Exception ex) { statusLabel.Text = "Ошибка
поиска"; MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error); }
finally { SetSearchState(false); }
}

private void SetSearchState(bool searching)
{
    searchButton.Enabled = !searching;
}

```

```

        searchBox.Enabled = !searching;
        progressBar.Visible = searching;
    }

    private async Task<List<Book>> SearchOpenLibrary(string
query, int limit)
    {
        var books = new List<Book>();
        try
        {
            string url =
"${OPEN_LIBRARY_SEARCH_URL}?q={Uri.EscapeDataString(query)}&limit
={limit}";

            var response = await httpClient.GetAsync(url);
            if (!response.IsSuccessStatusCode) return books;
            string json = await
response.Content.ReadAsStringAsync();
            using JsonDocument doc =
JsonDocument.Parse(json);
            if (doc.RootElement.TryGetProperty("docs", out
JsonElement docs))
                foreach (var el in docs.EnumerateArray())
                {
                    var book = ParseBookFromJson(el);
                    if (book != null
&& !string.IsNullOrEmpty(book.Title)) books.Add(book);
                }
        }
        catch { }
        return books;
    }

    private Book ParseBookFromJson(JsonElement el)
    {
        try
        {
            var book = new Book();
            if (el.TryGetProperty("key", out var k)) book.Key
= k.GetString();
            if (el.TryGetProperty("title", out var t))
book.Title = t.GetString();
            if (el.TryGetProperty("first_publish_year", out
var y) && y.ValueKind == JsonValueKind.Number)
book.FirstPublishYear = y.GetInt32();
            if (el.TryGetProperty("author_name", out var
authors) && authors.ValueKind == JsonValueKind.Array) { var list
= new List<string>(); foreach (var a in authors.EnumerateArray())
list.Add(a.GetString()); book.AuthorNames = string.Join(", ",
list); }
            if (el.TryGetProperty("cover_i", out var c) &&
c.ValueKind == JsonValueKind.Number) { book.CoverId =

```

```

c.GetInt32(); book.CoverUrl =
$"{COVER_IMAGE_URL}/id/{book.CoverId}-M.jpg"; }
        if (el.TryGetProperty("ratings_average", out var
r) && r.ValueKind == JsonValueKind.Number) book.Rating =
r.GetDouble();
        if (el.TryGetProperty("first_sentence", out var
fs) && fs.ValueKind == JsonValueKind.Array &&
fs.GetArrayLength() > 0) book.Description = fs[0].GetString();
        if (el.TryGetProperty("subject", out var
subjects) && subjects.ValueKind == JsonValueKind.Array)
{ book.Subjects = new List<string>(); foreach (var s in
subjects.EnumerateArray()) if (s.ValueKind ==
JsonValueKind.String) book.Subjects.Add(s.GetString()); }
        if (!string.IsNullOrEmpty(book.Key)) book.Url =
$"{OPEN_LIBRARY_BOOK_URL}{book.Key}";
        return book;
    }
    catch { return null; }
}

private async void
SearchResultsListView_SelectedIndexChanged(object sender,
EventArgs e)
{
    if (searchResultsListView.SelectedItems.Count == 0)
return;
    var idx = searchResultsListView.SelectedIndices[0];
    if (idx < 0 || idx >= currentBooks.Count) return;
    var book = currentBooks[idx];
    addToFavoritesButton.Enabled = true;
    bool already =
dbManager.IsBookInFavorites(currentUsername, book.Key ?? "");
    addToFavoritesButton.Text = already ? "✓ВЫЗБРАННОМ" :
"★ВЫЗБРАННОЕ";
    addToFavoritesButton.NormalColor = already ?
DrawingColor.LimeGreen : DrawingColor.Gold;
    dbManager.SaveViewHistory(currentUsername,
book.Key ?? "");
    var sb = new StringBuilder();
    sb.AppendLine($"📖 {book.Title}");
    sb.AppendLine();
    if (!string.IsNullOrEmpty(book.AuthorNames))
sb.AppendLine($"✍️Автор: {book.AuthorNames}");
    if (book.FirstPublishYear.HasValue)
sb.AppendLine($"📅Год: {book.FirstPublishYear}");
    if (book.Rating.HasValue) sb.AppendLine($"★Рейтинг:
{book.Rating:F1} / 5.0");
    if (contentFilterEnabled && !IsBookSafe(book))
sb.AppendLine("\n⚠️ВНИМАНИЕ: Книгасодержитконтент 18+");
}

```

```

        if (!string.IsNullOrEmpty(book.Description))
sb.AppendLine($"\\n📖 {book.Description}");
        if (!string.IsNullOrEmpty(book.Url))
        {
            sb.AppendLine($"\\n🔗 Открыть на OpenLibrary:
{book.Url}");
        }
        bookDetailsTextBox.Text = sb.ToString();
        await LoadBookCover(book);
    }

    private async Task LoadBookCover(Book book)
    {
        if (string.IsNullOrEmpty(book?.CoverUrl))
        { bookCoverPicture.Image = null; return; }
        var cachedCover =
dbManager.GetCachedCover(book.CoverId.ToString());
        if (cachedCover != null) { using var ms = new
MemoryStream(cachedCover); bookCoverPicture.Image =
DrawingImage.FromStream(ms); return; }
        try
        {
            var bytes = await
httpClient.GetByteArrayAsync(book.CoverUrl);
            dbManager.CacheCover(book.CoverId.ToString(),
bytes);

            using var ms = new MemoryStream(bytes);
            bookCoverPicture.Image =
DrawingImage.FromStream(ms);
        }
        catch { bookCoverPicture.Image = null; }
    }

    private void SearchResultsListView_DoubleClick(object
sender, EventArgs e)
    {
        if (searchResultsListView.SelectedItems.Count == 0)
return;
        var book =
currentBooks[searchResultsListView.SelectedIndices[0]];
        if (contentFilterEnabled && !IsBookSafe(book))
            if
(MessageBox.Show("Эта книга может содержать контент 18+. Вы уверены,
что хотите открыть её в браузере?", "Предупреждение",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning) !=
DialogResult.Yes) return;
        if (!string.IsNullOrEmpty(book.Url))
Process.Start(new ProcessStartInfo(book.Url) { UseShellExecute =
true });
    }

```

```

        private void AddToFavoritesButton_Click(object sender,
EventArgs e)
        {
            if (searchResultsListView.SelectedItems.Count == 0)
return;

            var book =
currentBooks[searchResultsListView.SelectedIndices[0]];
            if (contentFilterEnabled && !IsBookSafe(book))
                if (MessageBox.Show("Эта книга содержит контент 18+.
Вы уверены, что хотите добавить её в избранное?",
"Предупреждение", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning) != DialogResult.Yes) return;
            if (dbManager.IsBookInFavorites(currentUsername,
book.Key ?? "")) { MessageBox.Show("Книга уже в избранном!",
"Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);
return; }

            dbManager.SaveFavorite(currentUsername, book.Title,
book.AuthorNames ?? "Неизвестен",
book.FirstPublishYear?.ToString() ?? "?",
DateTime.Now.ToString("dd.MM.yyyy HH:mm"), book.Key ?? "",
book.CoverId?.ToString() ?? "", book.Rating, book.Description);
            addToFavoritesButton.Text = "✓В ИЗБРАННОМ";
            addToFavoritesButton.NormalColor =
DrawingColor.LimeGreen;
            RefreshStatsGraph();
            RefreshFavoritesCount();
            MessageBox.Show($"Книга '{book.Title}'
добавлена в избранное!", "Успех", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }

        private void LoadFavorites()
        {
            favoritesListView.Items.Clear();
            var favs = dbManager.LoadFavorites(currentUsername);
            foreach (var f in favs)
            {
                var item = new ListViewItem(new[]
{ f.Title.Length > 55 ? f.Title.Substring(0, 52) + "..." : f.Title,
f.Author.Length > 32 ? f.Author.Substring(0, 29) + "..." :
f.Author, f.Year, f.AddedDate }) { Tag = f };
                favoritesListView.Items.Add(item);
            }
        }

        private void
FavoritesListView_SelectedIndexChanged(object sender, EventArgs
e) => removeFromFavoritesButton.Enabled =
favoritesListView.SelectedItems.Count > 0;

```

```

        private void FavoritesListView_DoubleClick(object sender,
EventArgs e)
        {
            if (favoritesListView.SelectedItems.Count == 0)
return;
            var fav = favoritesListView.SelectedItems[0].Tag as
FavoriteBook;
            if (fav != null
&& !string.IsNullOrEmpty(fav.BookKey))
                Process.Start(new
ProcessStartInfo($"{OPEN_LIBRARY_BOOK_URL}{fav.BookKey}")
{ UseShellExecute = true });
        }

        private void RemoveFromFavoritesButton_Click(object
sender, EventArgs e)
        {
            if (favoritesListView.SelectedItems.Count == 0)
return;
            var keys = new List<string>();
            foreach (ListViewItem item in
favoritesListView.SelectedItems) if (item.Tag is FavoriteBook f)
keys.Add(f.BookKey);
            dbManager.RemoveFavorites(currentUsername, keys);
            LoadFavorites();
            RefreshFavoritesCount();
            RefreshStatsGraph();
            MessageBox.Show("Книги удалены из избранного!",
"Готово", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void ClearFavoritesButton_Click(object sender,
EventArgs e)
        {
            if (dbManager.CountUserFavorites(currentUsername) ==
0) { MessageBox.Show("В избранном пока пусто.", "Информация",
MessageBoxButtons.OK, MessageBoxIcon.Information); return; }
            if (MessageBox.Show("Очистить ВСЁ избранное?
Действие необратимо!", "Внимание", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning) == DialogResult.Yes)
            {
                dbManager.ClearUserFavorites(currentUsername);
                LoadFavorites();
                RefreshFavoritesCount();
                RefreshStatsGraph();
                MessageBox.Show("Избранное очищено!", "Готово",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }

        private void MainTabControl_SelectedIndexChanged(object
sender, EventArgs e)

```

```

        {
            if
(mainTabControl.SelectedTab?.Text.Contains("Профиль") == true)
            {
                RefreshFavoritesCount();
                UpdateProfileEmail();
                RefreshStatsGraph();
            }

            if
(mainTabControl.SelectedTab?.Text.Contains("Избранное") == true)
                LoadFavorites();

            if
(mainTabControl.SelectedTab?.Text.Contains("Рекомендации") ==
true && !recommendationsLoaded)
            {
                recommendationsLoaded = true;
                _ = LoadRecommendations();
            }
        }

        public void RefreshFavoritesCount() { if
(favoritesCountLabel != null) favoritesCountLabel.Text =
$"Книгвизбранном:
{dbManager.CountUserFavorites(currentUsername)}"; }

        private void ApplyNewTheme(string newTheme)
        {
            if (newTheme == currentTheme) return;
            currentTheme = newTheme;
            ApplyTheme();
            dbManager.UpdateUserTheme(currentUsername,
currentTheme);
            currentThemeLabel.Text = $"Тема: {currentTheme}";
            MessageBox.Show($"Темаизмененана «{currentTheme}»",
"Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void ApplyTheme()
        {
            if (!themes.ContainsKey(currentTheme)) currentTheme =
"Светлая";
            var scheme = themes[currentTheme];
            BackColor = scheme.Background;
            ForeColor = scheme.Foreground;
            UpdatePanelsOnly(Controls, scheme);
            UpdateLabelsOnly(Controls, scheme);
        }

        private void UpdatePanelsOnly(Control.ControlCollection
controls, ColorScheme scheme)

```

```

        {
            foreach (Control c in controls)
            {
                if (c is WinFormsPanel p && !(p is BookCard))
p.BackgroundColor = scheme.Background;
                else if (c is TabPage tp) tp.BackgroundColor =
scheme.Background;
                if (c.HasChildren && !(c is BookCard))
UpdatePanelsOnly(c.Controls, scheme);
            }
        }

        private void UpdateLabelsOnly(Control.ControlCollection
controls, ColorScheme scheme)
        {
            foreach (Control c in controls)
            {
                if (c is WinFormsLabel lbl && !(lbl.Parent is
BookCard) && !(lbl.Parent is ModernButton)) lbl.ForeColor =
scheme.Foreground;
                else if (c is WinFormsTextBox tb) { tb.BackgroundColor
= scheme.ControlBackground; tb.ForeColor = scheme.Foreground; }
                else if (c is RichTextBox rtb) { rtb.BackgroundColor =
scheme.ControlBackground; rtb.ForeColor = scheme.Foreground; }
                else if (c is ListView lv) { lv.BackgroundColor =
scheme.ControlBackground; lv.ForeColor = scheme.Foreground; }
                if (c.HasChildren && !(c is BookCard))
UpdateLabelsOnly(c.Controls, scheme);
            }
        }

        private void LogoutButton_Click(object sender, EventArgs
e)
        {
            if (MessageBox.Show("Выйти из аккаунта?",
"Подтверждение", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes) Close();
        }

        protected override void
OnFormClosing(FormClosingEventArgs e)
        {
            httpClient?.Dispose();
            dbManager?.Dispose();
            base.OnFormClosing(e);
        }

        public class Book
        {
            public string Key { get; set; }
            public string Title { get; set; }
            public string AuthorNames { get; set; }
        }

```

```

        public int? FirstPublishYear { get; set; }
        public int? CoverId { get; set; }
        public string CoverUrl { get; set; }
        public string Url { get; set; }
        public double? Rating { get; set; }
        public string Description { get; set; }
        public List<string> Subjects { get; set; }
    }

    public class FavoriteBook
    {
        public string Username { get; set; }
        public string Title { get; set; }
        public string Author { get; set; }
        public string Year { get; set; }
        public string AddedDate { get; set; }
        public string BookKey { get; set; }
        public string CoverId { get; set; }
    }

    public class ColorScheme
    {
        public DrawingColor Background { get; set; }
        public DrawingColor Foreground { get; set; }
        public DrawingColor ControlBackground { get; set; }
        public DrawingColor AccentColor { get; set; }
        public DrawingColor SecondaryColor { get; set; }
    }

    public static class StringExtensions
    {
        public static string Repeat(this string s, int count) =>
string.Concat(Enumerable.Repeat(s, count));
    }

    public static class EnumerableExtensions
    {
        public static IEnumerable<TSource> DistinctBy<TSource,
TKey>(this IEnumerable<TSource> source, Func<TSource, TKey>
keySelector)
        {
            var seenKeys = new HashSet<TKey>();
            foreach (var element in source) if
(seenKeys.Add(keySelector(element))) yield return element;
        }
    }

    static class Program
    {
        [STAThread]
        static void Main()
    }

```

```

        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
Application.Run(new LoginForm());
        }
    }
}

```

Приложение В

(обязательное)

Информационная система управления персональными подборками
литературных произведений

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		157

Презентация

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

ИНФОРМАЦИОННАЯ СИСТЕМА УПРАВЛЕНИЯ ПЕРСОНАЛЬНЫМИ ПОДБОРКАМИ ЛИТЕРАТУРНЫХ ПРОИЗВЕДЕНИЙ

АВТОР РАБОТЫ: СТУДЕНТ ГР.22ВВВ1

Саветкин Д.Д.

НАУЧНЫЙ РУКОВОДИТЕЛЬ: ДОЦЕНТ,
К.Т.Н. Бычков А.С.

Цель и функционал программы

Цель: разработка инструмента, автоматизирующего процесс поиска и первичной обработки библиографических метаданных.

Функционал :

- аутентификация и регистрация пользователя
- поиск библиографической информации
- просмотр детальной информации
- управление избранным
- возрастная фильтрация контента
- визуализация статистики
- экспорт библиографического списка

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.11.001 ПЗ

Лист

159

Язык и среда разработки

Среда разработки



Visual Studio 2022 - мощная интегрированная среда разработки, которая позволяет создавать разнообразные приложения для платформы Windows.

Язык разработки



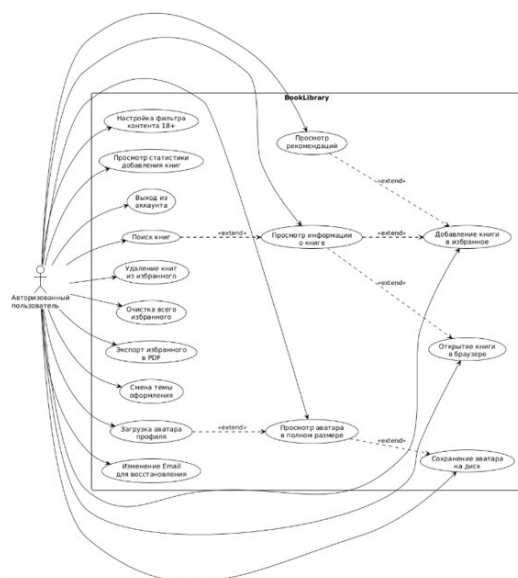
C# — это современный кроссплатформенный объектно-ориентированный язык программирования с открытым кодом для платформы разработчиков .NET с бесплатными инструментами

База данных



SQLite - это компактная и кроссплатформенная СУБД, которая хранит базу данных в единственном файле.

Варианты использования программы



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.11.001 ПЗ

Лист

160

Диаграмма классов и базы данных

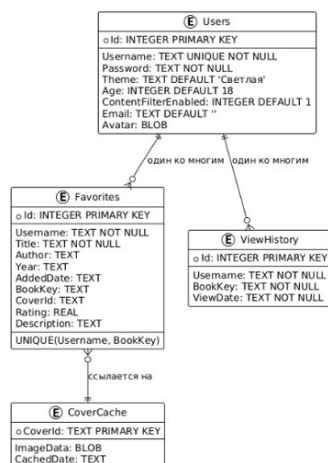


Диаграмма базы данных

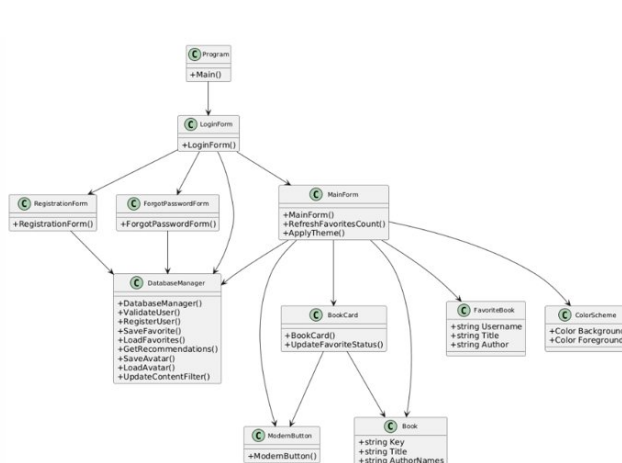
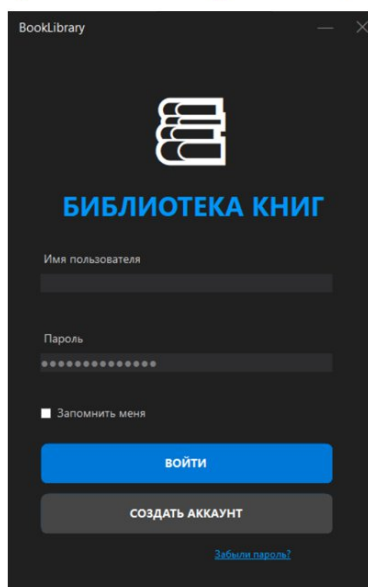
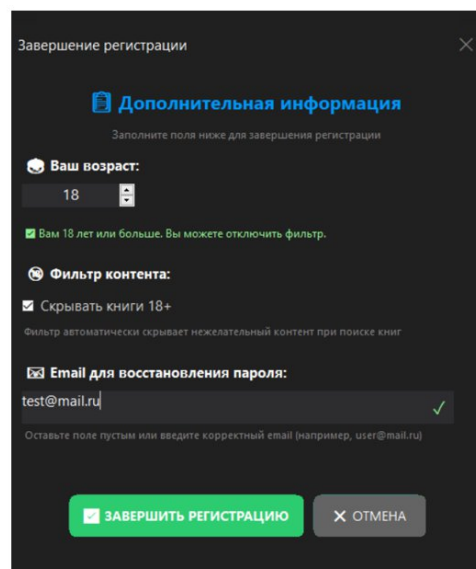
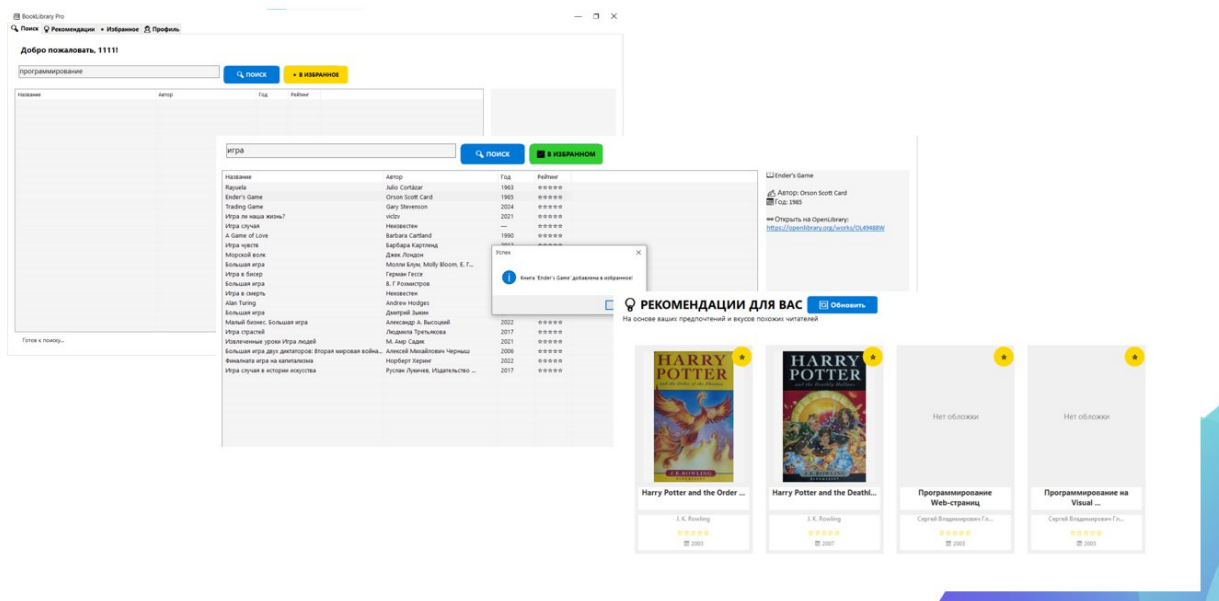


Диаграмма классов программы

Экран авторизации в приложении



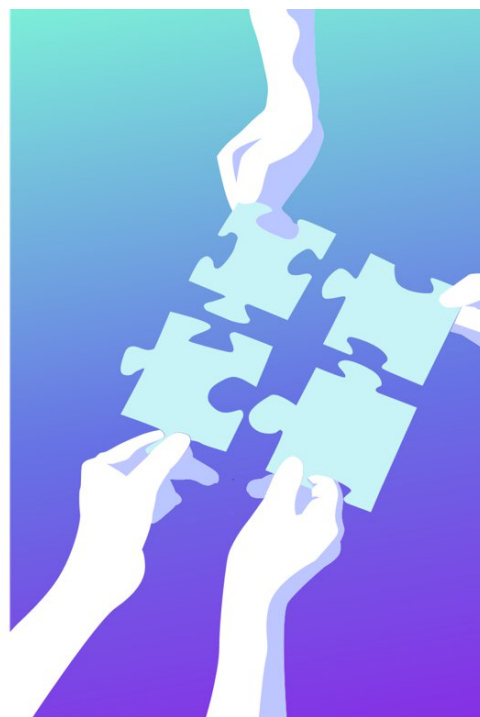
Главный экран после входа в аккаунт



ЗАКЛЮЧЕНИЕ

За период выполнения работы реализованы:

система регистрации и аутентификации пользователей с возможностью восстановления пароля через email, персонализированные профили с настраиваемыми темами оформления и возрастными ограничениями, поисковая система с фильтрацией запрещенного контента, по ключевым словам, система избранного с экспортом в PDF и визуальной статистикой чтения, а также рекомендательная система на основе коллаборативной фильтрации предпочтений пользователей.

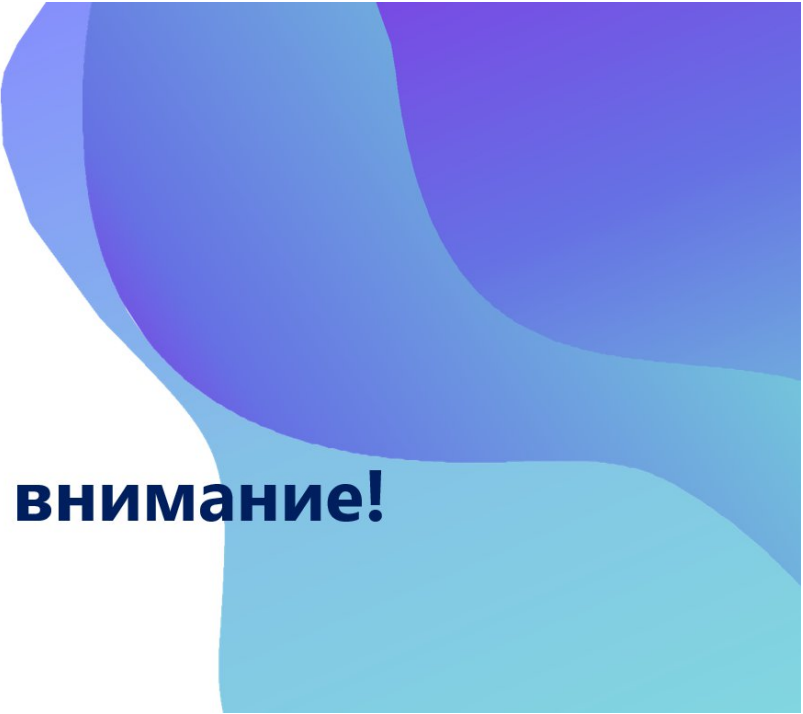


Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.11.001 ПЗ

Лист

162



Спасибо за внимание!

					ПГУ1.090301.11.001 ПЗ	Лист
						163
Изм.	Лист	№ докум.	Подпись	Дата		