Изображение выглядит как графическая вставка, Графика, мультфильм

Автоматически созданное описание

**КУРСОВОЙ ПРОЕКТ**

**По дисциплине: МДК 01.01 Разработка программных модулей**

**Тема: Разработка системы классов для приложения «Библиотека»**

**Специальность 09.02.07 «Информационные системы и программирование»**

|  |  |  |
| --- | --- | --- |
| **Выполнила студентка группы 302ИС-22** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **А. М. Пиголева** |
| **Руководитель** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Л. Б. Гусятинер** |

**Москва 2024**

Изображение выглядит как графическая вставка, Графика, мультфильм

Автоматически созданное описание

**УТВЕРЖДАЮ**

**Зам. директора КМПО**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.Ф. Гасанов**

**«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.**

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

**По дисциплине: МДК 01.01 Разработка программных модулей**

**Специальность 09.02.07 «Информационные системы и программирование»**

**Студент группы 302ИС-22 Анастасия Пиголева**

**ТЕМА: Разработка системы классов для приложения «Библиотека»**

Дата выдачи задания «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г.

Срок сдачи проекта «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г.

**Москва 2024**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc184228184)

[1. Описание предметной области 6](#_Toc184228185)

[1.1 Введение в предметную область 6](#_Toc184228186)

[1.2 Описание существующих разработок 10](#_Toc184228187)

[1.2.1 Koha 11](#_Toc184228188)

[1.2.2 ИРБИС 64+ 12](#_Toc184228189)

[2. Проектирование системы 15](#_Toc184228190)

[2.1 Общее назначение системы 15](#_Toc184228191)

[2.2 Требования к функциям, выполняемым системой. 16](#_Toc184228192)

[3. Реализация проекта системы 22](#_Toc184228193)

[3.1 Описание среды разработки 22](#_Toc184228194)

[3.1.2 Обоснование выбора инструментария по разработке 25](#_Toc184228195)

[3.2 Описание модулей. 26](#_Toc184228196)

[3.2.1 Модуль 1. Методы главного модуля 27](#_Toc184228197)

[3.2.2 Модуль 2. Модуль данных 29](#_Toc184228198)

[3.2.3 Модуль 3. Обработка данных 31](#_Toc184228199)

[3.2.4 Модуль 4. Модуль вспомогательных функций 34](#_Toc184228200)

[3.2.5 Модуль 5. Модуль файл данных 43](#_Toc184228201)

[3.3 Описание применения средства отладки 44](#_Toc184228202)

[3.4 To-Do Лист 45](#_Toc184228203)

[ЗАКЛЮЧЕНИЕ 46](#_Toc184228204)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 47](#_Toc184228205)

[ПРИЛОЖЕНИЕ 48](#_Toc184228206)

[Приложение 1 48](#_Toc184228207)

[Приложение 2 51](#_Toc184228208)

[Приложение 3 53](#_Toc184228209)

# **ВВЕДЕНИЕ**

Управление библиотекой является сложным процессом, требующим эффективного учета, хранения и обработки информации. В современном цифровом пространстве автоматизация библиотечных процессов становится важным условием успешного функционирования и развития учреждений. Организация доступа к библиотечным ресурсам требует постоянного мониторинга данных и их упорядочения.

Целью данного проекта является разработка системы классов для приложения «Библиотека», которая обеспечит учет, хранение и управление данными о книжном фонде, пользователях и их взаимодействиях с библиотекой. Для достижения этой цели необходимо выполнить следующие задачи:

1. проанализировать предметную область;
2. проанализировать готовые решения;
3. подготовить техническое задание;
4. подготовить план тестирования;
5. обосновать выбор инструментов и средств разработки;
6. описать реализацию технического задания;
7. выполнить тестирование.

Объект исследования – автоматизация библиотечных процессов, включая управление книжным фондом, учет выданных и возвращенных книг, регистрация читателей, а также список всех хранящихся данных библиотеки.

Предмет исследования – учет, хранение и обработка данных о книгах, читателях и их взаимодействиях с библиотекой.

Актуальность темы курсового проекта обусловлена необходимостью автоматизации процессов, связанных с управлением книжным фондом в библиотеках. Библиотека как учреждение, хранящее и предоставляющее доступ к культурным и образовательным ресурсам, нуждается в эффективных цифровых инструментах для обработки информации. Классические бумажные системы учета и обработки данных в библиотеках постепенно утрачивают свою эффективность. Они требуют значительных временных затрат, подвержены ошибкам и неудобны для анализа больших объемов данных. В условиях постоянного увеличения объема информации и числа пользователей библиотек, традиционные методы учета и обработки данных становятся неэффективными. Ручной учет, поиск книг по запросам, а также контроль за выдачей и возвратом книг требуют значительных трудозатрат и могут быть подвержены ошибкам. В связи с этим создание системы, автоматизирующей эти процессы, представляется необходимым шагом для повышения эффективности работы библиотеки. Кроме того, использование объектно-ориентированного подхода в разработке способствует созданию структурированного и легко расширяемого кода, что важно как с учебной, так и с практической точки зрения.

В первой части рассматривается предметная область, обосновывается целесообразность автоматизации и анализируются существующие решения.

Во второй части подробно описаны общее назначение системы, требования к функциям программы, которая должна их выполнять.

В третьей части будет рассказано про описание среды разработки, обоснование выбора инструментария по разработке, так же будет описаны модули, листинг и описание средств отладки вместе с ToDo-листом.

В заключительной части будут сделаны выводы по проделанной работе над проектом.

# **1. Описание предметной области**

## **Введение в предметную область**

Библиотечные системы сегодня играют важную роль в упрощении и автоматизации работы с книжными фондами. Они позволяют эффективно управлять информацией о книгах, пользователях и проводимых операциях. Такие системы востребованы не только в традиционных библиотеках, но и в образовательных учреждениях, организациях и даже частных коллекциях, где нужно упорядочивать доступ к большим объемам информации.

Ключевая задача библиотечной системы — сделать работу с книгами удобной и доступной для всех участников. Для читателей это возможность быстро находить нужные книги и отслеживать их статус, а для администраторов — вести учет фондов, контролировать выдачу и возврат литературы.

С развитием технологий библиотечные приложения становятся более гибкими. Они предлагают широкий функционал, начиная от простого поиска книг и заканчивая просмотром истории операций. Такие приложения помогают минимизировать ошибки, сократить время на выполнение задач и автоматизировать рутинные процессы.

Разработка системы классов для приложения «Библиотека» базируется на нескольких ключевых концепциях:

* Учет и управление ресурсами: Организация структуры данных о книжных фондах, авторах, издательствах и читателях.
* Клиентский сервис: Обеспечение удобного интерфейса для поиска, бронирования и выдачи книг.
* Анализ данных: Использование данных для мониторинга популярности книг, учета задолженностей и оценки работы библиотеки.
* Современные требования к библиотечным системам:
* Интеграция: Возможность подключения к внешним базам данных и каталогам.
* Мобильность: Поддержка работы через мобильные устройства для удобства пользователей.
* Безопасность: Обеспечение защиты персональных данных пользователей и информации о книжном фонде.
* Масштабируемость: Возможность адаптации системы под объем фонда и потребности библиотеки.

Библиотеки различаются по своим функциям, назначению и аудитории, для которой они созданы.

Существует большое количество видов библиотек:

1. Традиционные библиотеки

Эти библиотеки занимаются хранением и выдачей физических книг. Основной задачей является учет фондов и упрощение работы библиотекарей, включая каталогизацию книг, регистрацию читателей и контроль операций выдачи и возврата. Такие библиотеки часто используют автоматизированные системы для поиска книг по каталогу, отслеживания задолженностей и управления базой данных.

1. Электронные библиотеки

Предназначены для хранения и распространения цифрового контента, включая электронные книги, научные статьи, аудиокниги и мультимедиа. Такие библиотеки предоставляют удобный удаленный доступ через веб-платформы или мобильные приложения. Электронные библиотеки обеспечивают быструю доставку информации пользователям, функцию поиска по тексту, а также интеграцию с другими образовательными и исследовательскими сервисами.

1. Корпоративные библиотеки

Эти библиотеки создаются внутри организаций и компаний для хранения специализированных материалов, таких как внутренние документы, инструкции, учебные пособия или профессиональная литература. Они упрощают доступ сотрудников к важным ресурсам, способствуют обучению персонала и поддерживают процессы внутри компании.

1. Учебные библиотеки

Фокусируются на предоставлении учебных материалов для студентов и преподавателей. Чаще всего такие библиотеки имеют учебники, научные статьи, методические пособия, а также доступ к базам данных научных исследований. Современные учебные библиотеки объединяют традиционные и электронные ресурсы, позволяя пользователям выбирать между физическим посещением и удаленным доступом.

1. Персональные библиотеки

Создаются для управления частными коллекциями книг. Такие библиотеки ориентированы на индивидуальных пользователей, которым важно упорядочить свою коллекцию, отслеживать, какие книги они прочитали, или вести учет, кому были одолжены книги. Программы для персональных библиотек часто предоставляют функции добавления заметок, сортировки по жанрам и создания рекомендаций.

1. Специализированные библиотеки

Обслуживают узкоспециализированные аудитории. Это могут быть медицинские библиотеки для врачей, юридические — для юристов, технические — для инженеров и разработчиков. Такие библиотеки содержат редкие или специфические издания, часто включают архивы и материалы для научных исследований.

Методологии разработки программного обеспечения играют ключевую роль в успешной реализации проектов, так как они определяют структуру и последовательность этапов разработки. Для разработки системы классов приложения «Библиотека» выбрана итерационная модель, которая является гибкой и позволяет эффективно адаптировать систему к меняющимся требованиям в ходе разработки.

Итерационная модель разработки программного обеспечения предполагает разделение процесса на несколько итераций, каждая из которых включает в себя полный цикл разработки, от планирования до тестирования. Каждая итерация строится на предыдущей и позволяет постепенно улучшать систему, добавляя новые функциональные возможности и устраняя выявленные ошибки. Этот подход помогает гибко реагировать на изменения в требованиях и предоставляет возможность вовремя корректировать ход проекта.

Основные этапы итерационной модели:

1. Планирование: Определение цели текущей итерации, сбор и анализ требований, подготовка плана разработки.
2. Проектирование: Разработка архитектуры системы и проектирование структуры классов.
3. Реализация: Кодирование и реализация функций в рамках текущей итерации.
4. Тестирование: Проверка функциональности и исправление ошибок, выявленных на предыдущих этапах.
5. Оценка и улучшение: Оценка текущего состояния системы, анализ полученных результатов и корректировка плана для следующей итерации.

Преимущества итерационной модели:

* Гибкость: Позволяет адаптировать проект в процессе работы и вовремя учитывать изменения требований или улучшения.
* Раннее выявление проблем: Регулярное тестирование и оценка результатов каждой итерации позволяют оперативно выявлять и устранять ошибки.
* Участие пользователей: Возможность демонстрации промежуточных версий системы пользователям на каждом этапе помогает лучше учитывать их потребности.

Выбор итерационной модели для разработки системы классов «Библиотека» обусловлен необходимостью создания гибкой и адаптивной системы, которая будет постепенно улучшаться и корректироваться на основе отзывов пользователей и новых требований, что особенно важно при разработке сложных информационных систем.

## **Описание существующих разработок**

В данном разделе рассмотрены некоторые уже существующие системы для управления библиотеками, их цели и основные требования. Интерфейсы таких программ обычно просты и понятны, что делает их удобными как для библиотекарей, так и для читателей.

Цели использования системы управления библиотекой:

1. Автоматизация работы библиотекарей.
2. Учет книжного фонда и обеспечение быстрого поиска книг.
3. Регистрация пользователей и управление их учетными записями.
4. Контроль операций по выдаче и возврату книг.
5. Формирование отчетов: востребованность книг, частота посещений, задолженности.

Основные требования к таким системам:

* Соответствие отчетности и учету книжного фонда законодательным нормам и внутренним правилам библиотек.
* Удобный учет и хранение информации о книгах, авторах и читателях.
* Возможность отслеживания статуса книг (в наличии, на руках, возвращена).
* Интеграция с электронными базами данных и библиотечными каталогами.
* Возможность формирования истории операций для анализа деятельности библиотеки.

## **Koha**

Полное название — Koha автоматизированная библиотечная информационная система (АБИС)

Koha — это интегрированная библиотечная система с открытым исходным кодом, предназначенная для автоматизации всех библиотечных процессов: от учета книг до управления пользователями и отчетности. Она является одной из самых популярных систем с открытым исходным кодом в области управления библиотеками и используется множеством организаций по всему миру. Используется во всём мире общественными, школьными и специальными библиотеками, а также в некоторых крупных академических библиотеках.

Достоинства Koha:

1. Поддержка множества библиотечных процессов (каталогизация, учет пользователей, обработка заказов и т. д.)
2. Открытый исходный код, что позволяет адаптировать систему под конкретные нужды.
3. Многоязычный интерфейс, включая русский, что делает программу доступной для разных регионов.
4. Модульность — множество встроенных модулей для управления каталогами, закупками, отчетами и т. д.
5. Поддержка интеграции с внешними сервисами и базами данных.

Koha представляет собой мощную библиотечную систему с открытым исходным кодом, которая позволяет автоматизировать управление библиотечными процессами, такими как учет книг, пользователей и взаимодействие с другими библиотеками. Благодаря простому и понятному интерфейсу, Koha доступна как для опытных пользователей, так и для начинающих. Возможность модификации исходного кода делает систему универсальной и подходящей для разнообразных библиотек. Однако, при настройке и использовании системы могут возникнуть сложности, связанные с необходимостью технической настройки и адаптации под специфические нужды организации. Кроме того, для крупных библиотек работа с Koha может требовать дополнительных усилий по масштабированию и настройке, особенно если необходимо интегрировать систему с другими программными решениями или обрабатывать большое количество данных.

## **ИРБИС 64+**

ИРБИС 64+ — это расширенная версия отечественной автоматизированной библиотечно-информационной системы ИРБИС 64, разработанная государственной публичной научно-технической библиотекой России. Она предназначена для комплексной автоматизации всех библиотечных процессов, включая управление фондами, каталогизацией, выдачей, статистикой и отчетностью.

Достоинства ИРБИС 64+:

1. Расширенный функционал: поддержка множества модулей для учета различных типов ресурсов, включая электронные.
2. Интеграция с электронными библиотеками: возможность работы с удаленными коллекциями и базами данных.
3. Локальная и сетевая архитектура: подходит как для индивидуальных библиотек, так и для распределенных сетей.
4. Гибкость настройки: возможность адаптации интерфейса и функций под нужды конкретной библиотеки.

ИРБИС 64+ отличается высокой функциональностью, предлагая широкий набор инструментов для работы с библиотечными ресурсами и обеспечивая совместимость с международными стандартами (MARC, RUSMARC). Это делает систему особенно востребованной среди научных, образовательных и общественных учреждений, где требуется автоматизация сложных библиотечных процессов. ИРБИС 64+ позволяет вести каталогизацию различных типов документов, включая печатные издания, электронные ресурсы и мультимедийные материалы.

Кроме того, система поддерживает интеграцию с электронными библиотеками, что упрощает доступ к внешним информационным ресурсам и базам данных. Благодаря локальной архитектуре, ИРБИС 64+ не зависит от стабильности интернет-соединения, что делает её привлекательной для учреждений, работающих в условиях ограниченной инфраструктуры. Вместе с тем система предоставляет возможности для сетевой работы, что позволяет использовать её в распределённых библиотечных сетях. Однако, несмотря на преимущества, ИРБИС 64+ имеет и несколько недостатков. Cистема не всегда отличается простотой в использовании, что может потребовать дополнительного времени на обучение персонала. Несмотря на наличие функций для интеграции с внешними ресурсами, сама система может быть ограничена в плане гибкости при работе с нестандартными форматами данных или уникальными потребностями пользователей.

Анализ готовых решений показывает, что каждая из информационных систем для библиотек имеет свои уникальные особенности, преимущества и недостатки. Выбор подходящей системы зависит от специфических потребностей библиотеки, ее размера и бюджета. Важно учитывать не только функциональные возможности системы, но и стоимость, а также возможность интеграции с другими программами для достижения максимальной эффективности работы.

Таким образом, разработка системы классов для приложения «Библиотека» требует глубокого понимания как исторических, так и современных аспектов библиотечного дела. Введение в предметную область подчеркивает важность учета ресурсов, организации клиентского сервиса и анализа данных, что является основой для создания эффективных информационных систем. Современные требования к библиотечным системам, включая интеграцию, мобильность и безопасность, играют ключевую роль в выборе подходящего решения. Анализ готовых решений, таких как ИРБИС 64+ и Koha, демонстрирует разнообразие функционала и особенностей, что позволяет библиотекам выбирать системы, соответствующие их специфическим нуждам и бюджету. В конечном итоге, правильный выбор программного обеспечения может значительно повысить эффективность работы библиотеки и улучшить качество обслуживания пользователей.

# **2. Проектирование системы**

## **2.1 Общее назначение системы**

Система «Библиотека» предназначена для автоматизации процессов управления библиотечным фондом, улучшения взаимодействия между библиотекарями и читателями, а также повышения эффективности работы персонала. Данное программное обеспечение обеспечивает удобный доступ к информации о книгах, возможность бронирования и упрощает работу с отчетностью. Основные цели и экономические эффекты от внедрения системы включают:

* Ускорение процессов обслуживания: Автоматизация поиска, бронирования и выдачи книг позволяет сократить время ожидания читателей и повысить их удовлетворенность.
* Учет библиотечного фонда: Система предоставляет возможность точного отслеживания текущего состояния фонда, контроля за возвратом книг и управления инвентаризацией.
* Повышение точности учета: Электронный учет исключает ошибки, связанные с ручным ведением записей, что способствует сохранности фонда и повышению эффективности работы.
* Анализ данных и прогнозирование: Система позволяет собирать и анализировать данные о предпочтениях читателей, популярных книгах и статистике использования фонда, что помогает в планировании закупок и улучшении обслуживания.
* Улучшение взаимодействия с пользователями: Внедрение системы бронирования и уведомлений о сроках возврата книг помогает повысить комфорт и лояльность читателей.
* Снижение операционных затрат: Автоматизация рутинных задач, таких как поиск книг и учет, позволяет снизить нагрузку на персонал и повысить общую производительность.

Данная система предоставляет удобный интерфейс для поиска книг с подробными описаниями, возможностью фильтрации по жанрам и авторам, а также функцией сравнения изданий. Она также открывает возможности для интеграции с электронными библиотеками, проведения мероприятий и внедрения программ лояльности.

Система «Библиотека» может быть эффективно внедрена как в публичных, так и в корпоративных библиотеках, что способствует расширению читательской аудитории и повышению их вовлеченности. Более того, данное решение позволяет оптимизировать управление фондом и улучшить качество обслуживания читателей.

## **2.2 Требования к функциям, выполняемым системой.**

В результате исследования предметной области были выявлены следующие функции, которые должны быть реализованы в системе «Библиотека»:

* Добавление и удаление авторов и книг в библиотечный фонде.
* Регистрация новых читателей с проверкой корректности введенных данных (email, номер телефона).
* Выдача книг читателям и регистрация записей о выдаче, с возможностью возврата книги и учета даты, когда книга была выдана и возвращена.
* Обновление данных, занесенных в систему библиотечного фонда, актуализация и обработка данных библиотеки.
* Просмотр карточки читателя с историей выданных книг, его личной информации и созданием отчета в формате Word.
* Просмотр всех имеющихся книг в библиотечном фонде с указанием, кто является автором, жанр книги, год издания данной книги и ее ISBN.
* Обеспечение удобного пользовательского интерфейса для взаимодействия с системой.
* Просмотр всех имеющихся данных, которые занесены в библиотечный фонд.

Также в системе должен быть реализован графический интерфейс для удобного взаимодействия пользователей с приложением и база данных для надежного хранения и обработки информации о библиотечном фонде, читателях и выданных книгах.

Для продолжения разработки проекта и создания программы необходимо провести глубокий детальный анализ среды разработки. Также требуется подготовить диаграмму прецедентов и IDEF1X-диаграмму.

Определим основные объекты. Их описание представим в Таблице 1.

Таблица 1 – Описание основных объектов

|  |  |  |
| --- | --- | --- |
| № | Наименование объекта | Краткое описание |
| 1 | Книга | |  | | --- | | Информация о книгах, доступных в библиотеке |  |  | | --- | |  | |
| 2 | Автор | |  | | --- | | Информация об авторах книг |  |  | | --- | |  | |
| 3 | Читатель | |  | | --- | | Информация о читателях библиотеки |  |  | | --- | |  | |
| 4 | |  | | --- | |  |   Читательская карточка | |  | | --- | | Карточка, выдаваемая читателю с его личными данными и записью о выдаче и возврате книг |  |  | | --- | |  | |
| 5 | Библиотекарь | |  | | --- | | Информация о сотрудниках библиотеки, их рабочего графика |  |  | | --- | |  | |
| 6 | |  | | --- | | Запись о выдаче |  |  | | --- | |  | | Информация о книгах, которые были выданы читателям |

Сущность «Книга» состоит из 6 атрибутов. Каждой книге присваивается уникальный идентификатор (book\_id), название, идентификатор автора, жанр, год и ISBN. Уникальный идентификатор (book\_id) является первичным ключом. Идентификатор (author\_id) является внешним ключом.

Таблица 2 – атрибуты сущности «Книга»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Описание |
| book\_id | INTEGER PRIMARY KEY | Уникальный идентификатор книги |
| title | VARCHAR(255) | Название книги |
| author\_id | INTEGER | Идентификатор автора книги |
| genre | VARCHAR(255) | |  | | --- | |  |   Жанр книги |
| year | INTEGER | Год издания книги   |  | | --- | |  | |
| isbn | VARCHAR(255) | Международный стандартный номер книги (ISBN) |

Сущность «Автор» состоит 4 из атрибутов. Каждому автору присваивается уникальный идентификатор (author\_id), имя, год рождения, страна. Уникальный идентификатор (author\_id) является первичным ключом.

Таблица 3 – атрибуты сущности «Автор»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Описание |
| author\_id | INTEGER PRIMARY KEY | Уникальный идентификатор автора |
| name | VARCHAR(255) | Имя автора |
| birth\_year | INTEGER | Год рождения автора |
| country | VARCHAR(255) | Страна рождения автора |

Сущность «Читатель» состоит из 4 атрибутов. Каждому читателю присваивается уникальный идентификатор (reader\_id), имя читателя, электронная почта, номер телефона. Уникальный идентификатор (reader\_id) является первичным ключом.

Таблица 4 – атрибуты сущности «Читатель»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Описание |
| reader\_id | INTEGER PRIMARY KEY | Уникальный идентификатор читателя |
| name | VARCHAR(255) | |  | | --- | | Имя читателя |  |  | | --- | |  | |
| email | VARCHAR(255) | |  | | --- | | Электронная почта читателя |  |  | | --- | |  | |
| phone | VARCHAR(255) | Номер телефона читателя |

Сущность «Читательская карточка» состоит из 4 атрибутов. Каждой читательской карточке присваивается уникальный идентификатор (card\_id), идентификатор карточки, идентификатор читателя, дата выдачи карточки, дата окончания срока карточки. Уникальный идентификатор (card\_id) является первичным ключом. Идентификатор (reader\_id) является внешним ключом.

Таблица 5 – атрибуты сущности «Читательская карточка»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Описание |
| card\_id | INTEGER PRIMARY KEY | Уникальный идентификатор карточки |
| reader\_id | INTEGER | |  | | --- | | Идентификатор читателя |  |  | | --- | |  | |
| issue\_date | DATE | |  | | --- | | Дата выдачи карточки |  |  | | --- | |  | |
| expiration\_date | DATE | Дата окончания срока действия карточки |

Сущность «Библиотекарь» состоит из 5 атрибутов. Каждому библиотекарю присваивается уникальный идентификатор (librarian\_id), идентификатор библиотекаря, имя, почта, телефон, график работы. Уникальный идентификатор (librarian\_id) является первичным ключом.

Таблица 6 – атрибуты сущности «Библиотекарь»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Описание |
| librarian\_id | INTEGER PRIMARY KEY | Уникальный идентификатор библиотекаря |
| name | VARCHAR(255) | |  | | --- | |  |   Имя библиотекаря |
| email | VARCHAR(255) | |  |  | | --- | --- | | |  | | --- | |  | |   Электронная почта |
| phone | |  | | --- | | VARCHAR(255) |  |  | | --- | |  | | Номер телефона |
| schedule | |  | | --- | | VARCHAR(255) |  |  | | --- | |  | | График работы |

Сущность «Запись о выдаче» состоит из 6 атрибутов. Каждой записи о выдаче присваивается уникальный идентификатор записи (record\_id), идентификатор книги, выданной читателю, идентификатор читателя, идентификатор библиотекаря, выдавшего книгу, дата выдачи книги, дата возврата книги. Уникальный идентификатор (record\_id) является первичным ключом. Идентификаторы (reader\_id, book\_id, librarian\_id) являются внешними ключами.

Таблица 7 – атрибуты сущности «Запись о выдаче»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Описание |
| record\_id | INTEGER PRIMARY KEY | Уникальный идентификатор записи |
| book\_id | INTEGER | |  | | --- | | Идентификатор книги, выданной читателю | |
| reader\_id | INTEGER | |  | | --- | |  |   Идентификатор читателя |
| librarian\_id | |  | | --- | | INTEGER | | |  | | --- | |  |   Идентификатор библиотеки, выдавшей книгу |
| borrow\_date | |  | | --- | |  |   DATE | Дата выдачи книги |
| return\_date | DATE | Дата возврата книги |

После создания всех таблиц необходимо установить связи между ними.

Связь «Читатель» - «Читательская карточка».

Один читатель может иметь одну или несколько читательских карточек, но каждая карточка принадлежит только одному читателю. Связь «один-ко-многим».

Связь «Читатель» - «Запись о выдаче»

Один читатель может иметь несколько записей о выдаче книг, но каждая запись о выдаче принадлежит только одному читателю. Связь «один-ко-многим».

Связь «Запись о выдаче» - «Книга»

Каждая запись о выдаче может содержать только одну книгу, но одна книга может быть выдана несколько раз. Связь «один-ко-многим».

Связь «Запись о выдаче» - «Библиотекарь»

Один библиотекарь может оформить множество записей о выдаче, но каждая запись о выдаче создается только одним библиотекарем. Связь «один-ко-многим».

Связь «Книга» - «Автор»

Каждая книга имеет одного автора, но один автор может быть связан с несколькими книгами. Связь «один-ко-многим».

На основе выявленных сущностей для системы «Библиотека» будет создана IDEF1X-диаграмма, которая представлена на рисунке 4 в Приложении 2.

Рисунок 2 в Приложении 2 демонстрирует диаграмму прецедентов, которая визуализирует сценарии использования системы и показывает, как различные варианты взаимодействуют с функционалом библиотеки.

**2.4 План тестирования**

Основной целью тестирования автоматизированной системы «Библиотека» является проверка функциональности, производительности и удобства использования. Это поможет выявить любые ошибки или недостатки до начала эксплуатации системы, обеспечив высокое качество и надежность работы для пользователей, включая библиотекарей и читателей.

Объектами тестирования являются следующие процессы: регистрация читателя, просмотр книг, добавление и удаление авторов и книг в библиотечный фонд, просмотр карточки читателя с историей выданных книг и созданием отчета в формате Word, просмотр всех данных, а также функционал выдачи и возврата книг.

План тестирования приведен в Приложении 1.

Таким образом, в данной главе было подробно рассмотрено общее назначение системы «Библиотека», направленной на автоматизацию управления библиотечными процессами и улучшение обслуживания читателей. Выделены ключевые преимущества от внедрения системы, такие как повышение скорости обслуживания, улучшение учета книг и пользователей, а также использование анализа данных для оптимизации работы библиотеки. Эти аспекты подчеркивают важность системы для повышения эффективности работы учреждения.

Кроме того, описаны функции, которые должны быть реализованы в системе, включая, хранение информации о книгах, читателях и выданных экземплярах, включая возврат этих экземпляров, добавление и удаление книг и авторов, регистрацию пользователей, создание отчетов и обновление информации в системе, а также предоставление удобного интерфейса. Описаны сущности, их атрибуты и взаимосвязи, что позволяет понять структуру базы данных и ее функциональность.

Также был разработан план тестирования, охватывающий проверку функциональности и производительности системы. Это позволит выявить и устранить проблемы, обеспечив высокое качество работы для библиотекарей и читателей.

## **3. Реализация проекта системы**

## **3.1 Описание среды разработки**

Для разработки системы «Библиотека» программист должен обладать набором ключевых компетенций, необходимых для реализации всех требований проекта. В первую очередь требуется знание принципов объектно-ориентированного программирования (ООП), которое позволит создать продуманную архитектуру классов, обеспечивая удобство расширения и сопровождения системы.

Так же необходимо разбираться в проектировании и управлении базами данных, включая создание структур данных, написание запросов и их оптимизацию. Это обеспечивает эффективное хранение и обработку информации о библиотечном фонде, читателях и операциях с книгами.

Знание методов проектирования графического интерфейса пользователя необходимо для создания понятного и функционального интерфейса для пользователя, что повысит удобство работы пользователей с системой.

Для обеспечения стабильности и надежности работы системы требуются навыки тестирования и отладки, что позволит выявлять и устранять ошибки на различных этапах разработки.

Кроме того, программист должен владеть инструментами для управления версиями кода, что особенно важно при совместной разработке и дальнейшей поддержке системы. Наконец, программист должен быть способен документировать проект, создавая подробное описание системы, которое поможет в её эксплуатации и развитии.

**3.1.1 Выбор и описание программных инструментов**

При выборе инструментов разработки системы «Библиотека» были определены ключевые критерии.

Степень важности каждого критерия оценивалась по следующим уровням: низкая, ниже средней, средняя, выше средней, высокая. Представлена таблица 8.

Таблица 8 – Критерии выбора инструментов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерий | Простота сопровождения | Наличие библиотек | Понятный синтаксис | Скорость разработки | Универсальность языка |
| Важность критерия | Выше средней | Средняя | Высокая | Высокая | Средняя |

На основании этих критериев проведено сравнение языков программирования: C, Python и C++. Каждому языку присваивались оценки от 0 до 10 баллов за выполнение каждого критерия. Представлена таблица 9.

Таблица 9 – Сравнение языков программирования

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий/Язык программирования | C | Python | C++ |
| Простота сопровождения | 5 | 8 | 6 |
| Наличие библиотек | 7 | 9 | 8 |
| Понятный синтаксис | 4 | 9 | 5 |
| Скорость разработки | 3 | 9 | 4 |
| Универсальность языка | 8 | 8 | 9 |
| Итого баллов | 27 | 43 | 32 |

На основании полученных результатов для разработки системы был выбран язык Python, так как он обладает наиболее понятным синтаксисом, развитой экосистемой библиотек и высокой скоростью разработки. Его возможности позволяют эффективно реализовать поставленные задачи и упростить поддержку кода.

Таблица 10 – Критерии выбора среды разработки

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Критерий | Простота | Функционал | |  | | --- | | Удобство использования |  |  | | --- | |  | | |  | | --- | | Документация на русском языке |  |  | | --- | |  | |
| Важность критерия | Средняя | Высокая | Высокая | Ниже средней |

На основании критериев в таблице 10 проведено сравнение трех сред разработки: PyCharm, Visual Studio Code, Jupyter Notebook. Каждая среда оценивалась по шкале от 0 до 10. Представлена таблица 11.

Таблица 11 – Оценка сред разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий / Среда разработки | PyCharm | Visual Studio Code | Jupyter Notebook |
| Простота | 8 | 7 | 9 |
| Функционал | 9 | 7 | 6 |
| Удобство использования | 9 | 8 | 8 |
| Документация на русском языке | 6 | 5 | 4 |
| Итого баллов | 32 | 27 | 27 |

По результатам оценки для разработки системы была выбрана PyCharm, так как она предоставляет высокий функционал, удобство использования и лучше всего подходит для работы с Python.

Для разработки системы «Библиотека» были выбраны следующие библиотеки:

1. Библиотека: datetime – стандартная библиотека Python, предназначенная для работы с датами и временем. Она позволяет учитывать временные интервалы, регистрировать дату выдачи и возврата книг, а также автоматически определять срок действия карточки читателя.
2. Библиотека: python-docx – библиотека для работы с документами в формате Microsoft Word (DOCX). Она позволяет генерировать и редактировать текстовые документы, что используется для создания карточек читателей с личной информацией и историей выдачи книг.
3. Библиотека: re – стандартная библиотека Python для работы с регулярными выражениями. Она предоставляет инструменты для проверки корректности ввода данных, таких как адреса электронной почты и номера телефонов.

### **3.1.2** **Обоснование выбора инструментария по разработке**

На основании проведенного анализа и сравнений в таблицах 10 и 11 для разработки системы была выбрана среда разработки PyCharm. Программа получила наивысшую оценку благодаря своему богатому функционалу, удобству использования и наличию инструментов, оптимизированных для работы с языком Python.

Выбор языка программирования Python обоснован его универсальностью, широким спектром библиотек для работы, генерацией документов и удобным синтаксисом, что ускоряет процесс разработки. Python также обладает активным сообществом разработчиков, что упрощает поиск решений для возникающих задач. Это было так же проанализировано и сравнено в таблицах 8 и 9.

Для реализации функциональности системы «Библиотека» были выбраны следующие библиотеки, каждая из которых играет важную роль в обеспечении работы автоматизированной системы:

* 1. Библиотека: datetime – используется для работы с датами и временем. Обеспечивает учет сроков выдачи и возврата книг.
  2. Библиотека: python-docx – библиотека предназначена для работы с документами формата Word. Позволяет создавать карточки читателей, генерировать отчеты о выданных книгах и формировать документы, которые могут быть распечатаны или отправлены пользователям.
  3. Библиотека: re – Эта библиотека предназначена для работы с регулярными выражениями, что позволяет эффективно проверять и обрабатывать вводимые данные. Используется для проверки корректности формата email, номера телефона, что обеспечивает точность и надежность системы.

## **3.2 Описание модулей.**

Главный модуль – main.py. Отвечает за взаимодействие с пользователем, предоставляя текстовое меню для работы с системой. Включает функции для выполнения основных операций (Добавить автора, удалить автора, добавить книгу, удалить книгу, зарегистрировать читателя, просмотр карточки читателя, выдать книгу, вернуть книгу, просмотреть все книги, показать все данные, выход), координируя работу других модулей.

Модуль данных – models.py. Cодержит классы, которые определяют структуру данных в системе. Он включает следующие классы: Book, Author, Reader, LibraryCard, Librarian, BorrowRecord.

Модуль обработки данных – file\_handler.py. Содержит функции load\_data и save\_data, которые обеспечивают загрузку данных из файла (data.txt) и их сохранение. Используется для работы с внешними данными, необходимых для функционирования системы.

Модуль вспомогательных функций – functions.py. Включает набор универсальных функций, обеспечивающих выполнение ключевых операций.

Файл данных – data.txt. Хранит текущую информацию о системе: списки авторов, книг, читателей, библиотекарей и записей о выдаче книг. Используется модулем file\_handler для загрузки и сохранения данных.

## **3.2.1 Модуль 1. Методы главного модуля**

Данный модуль содержит функции для взаимодействия с пользователем через текстовое меню, позволяя выполнять различные операции с данными о библиотеке, такими как добавление и удаление авторов, книг, регистрация читателей, а также управление процессом выдачи и возврата книг. Он координирует работу с другими модулями, включая загрузку и сохранение данных.

Полный код проекта можно будет посмотреть в приложении 3.

Листинг 1. Функция main\_menu

def main\_menu():

authors, books, readers, librarians, borrow\_records = load\_data(DATA\_FILE)

while True:

print("-" \* 40)

print("Библиотека")

print("-" \* 40)

print("\nМеню:")

print("1. Добавить автора")

print("2. Удалить автора")

print("3. Добавить книгу")

print("4. Удалить книгу")

print("5. Зарегистрировать читателя")

print("6. Просмотр карточки читателя")

print("7. Выдать книгу")

print("8. Вернуть книгу")

print("9. Просмотреть все книги")

print("10. Показать все данные")

print("0. Выход")

choice = input("Выберите действие: ")

if choice == "1":

add\_author(authors)

elif choice == "2":

remove\_author(authors)

elif choice == "3":

add\_book(books, authors)

elif choice == "4":

remove\_book(books)

elif choice == "5":

register\_reader(readers)

elif choice == "6":

show\_reader\_card(readers, borrow\_records)

elif choice == "7":

lend\_book(books, readers, librarians, borrow\_records)

elif choice == "8":

return\_book(borrow\_records)

elif choice == "9":

show\_all\_books(books)

elif choice == "10":

show\_all\_data(books, authors, readers, librarians, borrow\_records)

elif choice == "0":

save\_data(authors, books, readers, librarians, borrow\_records)

print("Работа завершена. Итоговые данные сохранены.")

break

else:

print("Неверный выбор. Попробуйте снова.")

## **3.2.2 Модуль 2. Модуль данных**

Данный модуль представляет собой реализацию объектов данных, которые используются в библиотечной системе. Он включает описание классов для представления ключевых сущностей: книг, авторов, читателей, библиотекарей, библиотечных карточек и записей о выдаче книг. Эти классы обеспечивают структуру данных и хранят информацию, необходимую для управления библиотекой.

Листинг 2. Класс Book

class Book:

def \_\_init\_\_(self, title, author, genre, year, isbn):

self.title = title

self.author = author

self.genre = genre

self.year = year

self.isbn = isbn

Листинг 3. Класс Author

class Author:

def \_\_init\_\_(self, name, birth\_year, country):

self.name = name

self.birth\_year = birth\_year

self.country = country

Листинг 4. Класс Reader

class Reader:

def \_\_init\_\_(self, name, reader\_id, email, phone):

self.name = name

self.reader\_id = reader\_id

self.email = email

self.phone = phone

Листинг 5. Класс LibraryCard

class LibraryCard:

def \_\_init\_\_(self, reader, issue\_date, expiration\_date):

self.reader = reader

self.issue\_date = issue\_date

self.expiration\_date = expiration\_date

Листинг 6. Класс Librarian

class Librarian:

def \_\_init\_\_(self, name, librarian\_id, email, phone, schedule):

self.name = name

self.librarian\_id = librarian\_id

self.email = email

self.phone = phone

self.schedule = schedule

Листинг 7. Класс BorrowRecord

class BorrowRecord:

def \_\_init\_\_(self, book, reader, librarian, borrow\_date, return\_date=None):

self.book = book

self.reader = reader

self.librarian = librarian

self.borrow\_date = borrow\_date

self.return\_date = return\_date

## **3.2.3 Модуль 3. Обработка данных**

Данный модуль отвечает за загрузку и сохранение данных о книгах, авторах, читателях и библиотекарях из/в файл. Он включает функции для чтения данных из файла формата .txt и их записи обратно в файл в структуре, понятной программе. Модуль взаимодействует с другими частями программы, такими как классы Author, Book, Reader, Librarian и BorrowRecord, а также использует их для создания объектов данных, необходимых для работы библиотеки.

Листинг 8. Функция load\_data

def load\_data(filename):

authors, books, readers, librarians, borrow\_records = [], [], [], [], []

with open(filename, "r", encoding="utf-8") as file:

section = None

for line in file:

line = line.strip()

if line.startswith("[") and line.endswith("]"):

section = line[1:-1]

elif line and section:

if section == "Authors":

name, birth\_year, country = line.split(",")

authors.append(Author(name, int(birth\_year), country))

elif section == "Books":

title, author, genre, year, isbn = line.split(",")

books.append(Book(title, author, genre, int(year), isbn))

elif section == "Readers":

name, reader\_id, email, phone = line.split(",")

readers.append(Reader(name, int(reader\_id), email, phone))

elif section == "Librarians":

name, librarian\_id, email, phone, schedule = line.split(",")

librarians.append(Librarian(name, int(librarian\_id), email, phone, schedule))

elif section == "BorrowRecords":

title, reader\_name, librarian\_name, borrow\_date, return\_date = line.split(",")

book = next((b for b in books if b.title == title), title)

reader = next((r for r in readers if r.name == reader\_name), reader\_name)

librarian = next((l for l in librarians if l.name == librarian\_name), librarian\_name)

borrow\_records.append(BorrowRecord(book, reader, librarian, borrow\_date, return\_date if return\_date != "None" else None))

return authors, books, readers, librarians, borrow\_records

Листинг 9. Функция save\_data

def save\_data(authors, books, readers, librarians, borrow\_records):

filename = "library\_data.txt"

with open(filename, "w", encoding="utf-8") as file:

file.write("[Authors]\n")

for author in authors:

file.write(f"{author.name},{author.birth\_year},{author.country}\n")

file.write("\n[Books]\n")

for book in books:

file.write(f"{book.title},{book.author},{book.genre},{book.year},{book.isbn}\n")

file.write("\n[Readers]\n")

for reader in readers:

file.write(f"{reader.name},{reader.reader\_id},{reader.email},{reader.phone}\n")

file.write("\n[Librarians]\n")

for librarian in librarians:

file.write(

f"{librarian.name},{librarian.librarian\_id},{librarian.email},{librarian.phone},{librarian.schedule}\n")

file.write("\n[BorrowRecords]\n")

for record in borrow\_records:

return\_date = record.return\_date if record.return\_date else "Книга не возвращена"

file.write(f"{record.book.title if isinstance(record.book, Book) else record.book},"

f"{record.reader.name if isinstance(record.reader, Reader) else record.reader},"

f"{record.librarian.name if isinstance(record.librarian, Librarian) else record.librarian},"

f"{record.borrow\_date},{return\_date}\n")

print(f"Итоговый файл сохранен: {filename}")

## **3.2.4 Модуль 4. Модуль вспомогательных функций**

Данный модуль предоставляет функции для управления библиотечной системой: добавление авторов, книг, читателей с сохранением соответствующих данных; удаление авторов и книг из базы; проверка форматов введенных данных (email и номера телефона); регистрация читателей и управление их карточками, включая создание Word-файлов с информацией о читателях и их истории взятых книг; выдача и возврат книг с обновлением записей в журнале; отображение информации обо всех элементах библиотеки, включая авторов, книги, читателей, библиотекарей и записи о выдаче. Функции модуля позволяют взаимодействовать с данными через консоль или автоматизировать определенные процессы в библиотеке.

Листинг 10. Функция add\_author

def add\_author(authors):

name = input("Введите имя автора: ")

birth\_year = input("Введите год рождения автора: ")

country = input("Введите страну автора: ")

authors.append(Author(name, int(birth\_year), country))

print(f"Автор {name} добавлен.")

Листинг 11. Функция remove\_author

def remove\_author(authors):

name = input("Введите имя автора для удаления: ")

for author in authors:

if author.name.lower() == name.lower():

authors.remove(author)

print(f"Автор {name} удален.")

return

print("Автор не найден.")

Листинг 12. Функция add\_book

def add\_book(books, authors):

title = input("Введите название книги: ")

author\_name = input("Введите имя автора: ")

genre = input("Введите жанр книги: ")

year = input("Введите год издания книги: ")

isbn = input("Введите ISBN книги: ")

author = next((a for a in authors if a.name.lower() == author\_name.lower()), None)

if not author:

print(f"Автор {author\_name} не найден. Сначала добавьте автора.")

return

books.append(Book(title, author.name, genre, int(year), isbn))

print(f"Книга '{title}' добавлена.")

Листинг 13. Функция remove\_book

def remove\_book(books):

title = input("Введите название книги для удаления: ")

for book in books:

if book.title.lower() == title.lower():

books.remove(book)

print(f"Книга '{title}' удалена.")

return

print("Книга не найдена.")

Листинг 14. Функция register\_reader

def register\_reader(readers):

name = input("Введите имя читателя: ")

reader\_id = len(readers) + 1

email = input("Введите email читателя: ")

phone = input("Введите телефон читателя: ")

readers.append(Reader(name, reader\_id, email, phone))

print(f"Читатель {name} зарегистрирован.")

Листинг 15. Функция show\_reader\_card

def show\_reader\_card(readers, borrow\_records):

reader\_id = int(input("Введите ID читателя: ")) # Запрашиваем ID читателя

reader = next((r for r in readers if r.reader\_id == reader\_id), None)

if reader:

print("\nКарточка читателя:")

print(f" Имя: {reader.name}")

print(f" Email: {reader.email}")

print(f" Телефон: {reader.phone}")

print(f" ID читателя: {reader.reader\_id}")

card\_expiry = datetime(2027, 12, 31)

print(f"Срок действия карты: до {card\_expiry.strftime('%d.%m.%Y')}")

borrowed\_books = [record for record in borrow\_records if record.reader == reader]

if borrowed\_books:

print("\nИстория взятых книг:")

for record in borrowed\_books:

return\_date = record.return\_date if record.return\_date else "Не возвращена"

print(f" Книга: {record.book.title} — Автор: {record.book.author}, Жанр: {record.book.genre}, Год: {record.book.year}")

print(f" Дата выдачи: {record.borrow\_date}")

print(f" Дата возврата: {return\_date}")

print("-" \* 40)

generate\_reader\_card\_file(reader, borrowed\_books)

else:

print("\nУ этого читателя нет взятых книг.")

else:

print("Читатель с таким ID не найден.")

Листинг 16. Функция lend\_book

def lend\_book(books, readers, librarians, borrow\_records):

book\_title = input("Введите название книги: ")

book = next((b for b in books if b.title == book\_title), None)

if not book:

print("Книга не найдена.")

return

reader\_id = input("Введите ID читателя: ")

reader = next((r for r in readers if str(r.reader\_id) == reader\_id), None)

if not reader:

print("Читатель с таким ID не найден.")

return

librarian\_id = input("Введите ID библиотекаря: ")

librarian = next((l for l in librarians if str(l.librarian\_id) == librarian\_id), None)

if not librarian:

print("Библиотекарь с таким ID не найден.")

return

for record in borrow\_records:

if record.book.title == book\_title and record.reader.reader\_id == reader.reader\_id and not record.return\_date:

print(f"Ошибка: Книга '{book\_title}' уже выдана читателю с ID {reader.reader\_id}.")

return

borrow\_date = input("Введите дату выдачи книги (ГГГГ-ММ-ДД): ")

borrow\_records.append(BorrowRecord(book, reader, librarian, borrow\_date))

print(f"Книга '{book\_title}' успешно выдана читателю {reader.name}.")

Листинг 17. Функция return\_book

def return\_book(borrow\_records):

book\_title = input("Введите название возвращаемой книги: ")

reader\_id = input("Введите ID читателя: ")

for record in borrow\_records:

if record.book.title == book\_title and str(record.reader.reader\_id) == reader\_id and not record.return\_date:

return\_date = input("Введите дату возврата книги (ГГГГ-ММ-ДД): ")

record.return\_date = return\_date

print(f"Книга '{book\_title}' успешно возвращена читателем {record.reader.name}.")

return

print("Ошибка: Запись о выдаче книги не найдена или книга уже была возвращена.")

Листинг 18. Функция show\_all\_books

def show\_all\_books(books):

if not books:

print("В библиотеке пока нет книг.")

return

print("\nСписок всех книг в библиотеке:")

for i, book in enumerate(books, start=1):

print(f"{i}. {book.title} — Автор: {book.author}, Жанр: {book.genre}, Год: {book.year}, ISBN: {book.isbn}")

Листинг 19. Функция show\_all\_data

def show\_all\_data(books, authors, readers, librarians, borrow\_records):

print("\nАвторы:")

for author in authors:

print(f" {author.name} ({author.birth\_year}) - {author.country}")

print("\nКниги:")

for book in books:

print(f" {book.title} ({book.year}) - {book.author} [ISBN: {book.isbn}]")

print("\nЧитатели:")

for reader in readers:

print(f" {reader.name} (ID: {reader.reader\_id}) - {reader.email}, {reader.phone}")

print("\nБиблиотекари:")

for librarian in librarians:

print(f" {librarian.name} (ID: {librarian.librarian\_id}) - {librarian.email}, {librarian.phone}, {librarian.schedule}")

print("\nЗаписи о выдаче:")

for record in borrow\_records:

book\_title = record.book.title if isinstance(record.book, Book) else record.book

reader\_name = record.reader.name if isinstance(record.reader, Reader) else record.reader

librarian\_name = record.librarian.name if isinstance(record.librarian, Librarian) else record.librarian

status = f"Возвращена: {record.return\_date}" if record.return\_date else "На руках"

print(f" {book\_title} -> {reader\_name}, выдана: {librarian\_name} ({status})")

Листинг 20. Функция validate\_email

def validate\_email(email):

pattern = r"^[\w\.-]+@[\w\.-]+\.\w+$"

return re.match(pattern, email)

Листинг 21. Функция validate\_phone

def validate\_phone(phone):

return phone.isdigit()

Листинг 22. Функция register\_reader

def register\_reader(readers):

name = input("Введите имя читателя: ")

reader\_id = input("Введите ID читателя: ")

email = input("Введите email читателя: ")

while not validate\_email(email):

print("Неверный формат email. Попробуйте снова.")

email = input("Введите email читателя: ")

phone = input("Введите номер телефона читателя: ")

while not validate\_phone(phone):

print("Номер телефона должен состоять только из цифр. Попробуйте снова.")

phone = input("Введите номер телефона читателя: ")

readers.append(Reader(name, int(reader\_id), email, phone))

print("Читатель успешно зарегистрирован.")

Листинг 23. Функция generate\_reader\_card\_file

def generate\_reader\_card\_file(reader, borrowed\_books):

doc = Document()

doc.add\_heading("Карточка читателя", level=1)

doc.add\_paragraph(f"Имя: {reader.name}")

doc.add\_paragraph(f"Email: {reader.email}")

doc.add\_paragraph(f"Телефон: {reader.phone}")

doc.add\_paragraph(f"ID читателя: {reader.reader\_id}")

card\_expiry = datetime(2027, 12, 31)

doc.add\_paragraph(f"Срок действия карты: до {card\_expiry.strftime('%d.%m.%Y')}")

doc.add\_heading("История взятых книг", level=2)

if borrowed\_books:

for record in borrowed\_books:

return\_date = record.return\_date if record.return\_date else "Не возвращена"

doc.add\_paragraph(f"Книга: {record.book.title}")

doc.add\_paragraph(f" Автор: {record.book.author}")

doc.add\_paragraph(f" Жанр: {record.book.genre}")

doc.add\_paragraph(f" Год: {record.book.year}")

doc.add\_paragraph(f" Дата выдачи: {record.borrow\_date}")

doc.add\_paragraph(f" Дата возврата: {return\_date}")

doc.add\_paragraph("-" \* 40)

else:

doc.add\_paragraph("У этого читателя нет взятых книг.")

filename = f"Читатель\_{reader.reader\_id}\_{reader.name.replace(' ', '\_')}.docx"

doc.save(filename)

print(f"Файл карточки читателя сохранен как {filename}.")

## **3.2.5 Модуль 5. Модуль файл данных**

Данный набор данных представляет собой библиотечный каталог, структурированный по секциям.

Листинг 24. Файл с данными

[Authors]

Александр Пушкин,1799,Россия

Лев Толстой,1828,Россия

Фёдор Достоевский,1821,Россия

Джоан Роулинг,1965,Великобритания

Анна Джейн,1987,Россия

Стефани Майер,1973,США

Сара Дж.Маас,1986,США

[Books]

Евгений Онегин,Александр Пушкин,Роман,1833,978-5-699-12011-8

Война и мир,Лев Толстой,Роман,1869,978-5-389-07499-0

Преступление и наказание,Фёдор Достоевский,Роман,1866,978-5-17-087141-6

Гарри Поттер и философский камень,Джоан Роулинг,Фэнтези,1997,978-5-17-118366-2

Гарри Поттер и тайная комната,Джоан Роулинг,Фэнтези,1998,978-5-17-118367-9

Гарри Поттер и узник Азкабана,Джоан Роулинг,Фэнтези,1999,978-5-17-118368-6

Неогранённые алмазы,Анна Джейн,Роман,2016,978-5-699-84376-5

Тысяча сияющих солнц,Анна Джейн,Роман,2017,978-5-699-98865-6

Восхитительная ведьма,Анна Джейн,Роман,2022,978-5-00154-854-6

"Сумерки",Стефани Майер,Фэнтези,2005,978-5-17-057028-9

Новолуние,Стефани Майер,Фэнтези,2006,978-5-17-057029-6

Королевство шипов и роз,Сара Дж.Маас,Фэнтези,2015,978-5-17-104972-2

Империя бурь,Сара Дж.Маас,Фэнтези,2016,978-5-17-118442-3

[Readers]

Иван Иванов,1,ivanov@mail.ru,89001234567

Мария Петрова,2,petrova@mail.ru,89007654321

Алексей Смирнов,3,smirnov@mail.ru,89006543210

Елена Соколова,4,sokolova@mail.ru,89007778899

Анастасия Романова,5,romanova@mail.ru,89007557324

[Librarians]

Анна Смирнова,101,anna.smirnova@mail.ru,89005557788,Пн-Пт 9:00-18:00

Олег Кузнецов,102,oleg.kuznetsov@mail.ru,89004446655,Пн-Пт 10:00-19:00

[BorrowRecords]

Гарри Поттер и философский камень,Иван Иванов,Анна Смирнова,2024-11-01,None

Неогранённые алмазы,Мария Петрова,Олег Кузнецов,2024-11-10,None

Сумерки,Алексей Смирнов,Анна Смирнова,2024-11-15,None

Королевство шипов и роз,Елена Соколова,Олег Кузнецов,2024-11-20,None

Восхитительная ведьма,Анастасия Романова,Анна Смирнова,2024-12-12,None

# **3.3 Описание применения средства отладки**

В этом разделе демонстрируется умение использовать средства отладки.

В процессе написания проекта была получены две ошибки при попытке запуска программы:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1 – До применения средств отладки

После возникновения этой ошибки немедленно был произведён анализ. В результате была просмотрена 69-я и 21-я строка модуля main, что позволило идентифицировать проблему.

Внесённые изменения устранили проблему, и при повторном запуске программы ошибка не возникла. Это подтвердило, что решение оказалось корректным.

# **3.4 To-Do Лист**

В дальнейшем в данном проекте хотелось бы реализовать:

* 1. Базу данных.
  2. Графический интерфейс.
  3. Рекомендации на основе предпочтений читателей.
  4. Система уведомлений о сроках возврата.

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы была создана программная система «Библиотека», предназначенная для управления данными о книгах, авторах, читателях, библиотекарях и записях о выдаче книг.

В процессе работы были проведены исследования существующих библиотечных систем для анализа их функциональности и требований, а также изучена тема «Объектно-ориентированное программирование в Python» и работа с файловыми форматами для хранения данных. Особое внимание было уделено реализации системы классов и методов, обеспечивающих обработку данных.

Разработанная программа успешно выполняет все требования, указанные в техническом задании. Она позволяет добавлять, удалять и изменять данные о книгах и пользователях, вести учет выдачи и возврата книг, а также загружать и сохранять данные для обеспечения сохранности информации. Система протестирована и готова к использованию.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Интернет-источники

1. Документация Python // <https://pydocs.ru/>
2. Документация библиотеки datetime по использованию // <https://pydocs.ru/python-3-datetime/>
3. Документация библиотеки re по использованию // <https://pythonru.com/osnovy/modul-re-dlja-reguljarnyh-vyrazhenij-v-python?ysclid=m48nuulry412366286>
4. Документация библиотеки Document по использованию // <https://pydocs.ru/python-docx/>
5. Руководство по работе с PyCharm: установка, написание кода, отладка и профессиональные функции // <https://www.codeflow.site/ru/article/pycharm-guide>
6. Работа с текстовыми файлами в Python // <https://pythonru.com/osnovy/fajly-v-python-vvod-vyvod?ysclid=m48o8n3t2k247838857>
7. Виды автоматизированных библиотек // <https://lala.lanbook.com/sistemy-avtomatizacii-dlya-bibliotek?ysclid=m48oad1wer746022813>
8. Руководство по написанию «чистого» и читаемого кода на Python // <https://egorovegor.ru/python-pep8>
9. Понятие автоматизированной библиотечной системы и ее архитектуры. Принципы построения. // <https://ru.ruwiki.ru/wiki/Автоматизированная_библиотечная_информационная_система>

Стандарты и законодательные материалы

1. "ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления".
2. ГОСТ 9327-60 Бумага и изделия из бумаги.

Учебники и учебные пособия

1. Колкова, Н. И. Информационное обеспечение автоматизированных библиотечно-информационных систем (АБИС) : учебник для вузов / Н. И. Колкова, И. Л. Скипор. — 2-е изд. — Москва : Издательство Юрайт, 2024.
2. Алешин Л.И. АВТОМАТИЗАЦИЯ В БИБЛИОТЕКЕ. — Учебное пособие. Часть 1. — М.: Изд-во МГУКИ, ИПО Профиздат, 200 2 — 176 с.
3. Автоматизация библиотечно-библиографических процессов БАН: история и современность М.А. Белинская, И.И. Новицкая, Е.А. Тилева.
4. Севрюкова, А. А. Автоматизированные библиотечные системы / А. А. Севрюкова. — Текст : непосредственный // Молодой ученый. — 2015. — № 13 (93). — С. 857-859.

# **ПРИЛОЖЕНИЕ**

# *Приложение 1*

*План тестирования*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Идентификатор | Приоритет | Связанное требование | Модуль и подмодуль | Заголовок | Исходные данные | Шаги | Ожидаемый результат |
| |  | | --- | | LIB\_01 |  |  | | --- | |  | | A | |  | | --- | | Добавление автора |  |  | | --- | |  | | |  | | --- | | Управление авторами |  |  | | --- | |  | | |  | | --- | | Добавление нового автора |  |  | | --- | |  | | |  | | --- | | Необходимо добавить нового автора в систему. |  |  | | --- | |  | | |  | | --- | | Ввести имя, год рождения, страну автора. |  |  | | --- | |  | | |  | | --- | | Автор успешно добавлен в список авторов. |  |  | | --- | |  | |
| LIB \_02 | A | |  | | --- | | Удаление автора |  |  | | --- | |  | | |  | | --- | | Управление авторами |  |  | | --- | |  | | |  | | --- | | Удаление существующего автора |  |  | | --- | |  | | |  | | --- | | В текстовом файле есть автор, который может быть удален. |  |  | | --- | |  | | |  | | --- | | Ввести имя автора для удаления. |  |  | | --- | |  | | |  | | --- | | Автор удален из текстового файла. |  |  | | --- | |  | |
| LIB \_03 | A | |  | | --- | | Добавление книги |  |  | | --- | |  | | |  | | --- | | Управление книгами |  |  | | --- | |  | | |  | | --- | | Добавление новой книги |  |  | | --- | |  | | |  | | --- | | Необходимо добавить книгу с данными об авторе. |  |  | | --- | |  | | |  | | --- | | Ввести название книги, имя автора, жанр, год и ISBN. |  |  | | --- | |  | | |  | | --- | | Книга успешно добавлена в текстовый файл. |  |  | | --- | |  | |
| LIB \_04 | A | |  | | --- | | Удаление книги |  |  | | --- | |  | | |  | | --- | | Управление книгами |  |  | | --- | |  | | |  | | --- | | Удаление существующей книги |  |  | | --- | |  | | В текстовом файле есть книга, которая может быть удалена. | |  | | --- | | Ввести название книги для удаления. |  |  | | --- | |  | | |  | | --- | | Книга удалена из текстового файла. |  |  | | --- | |  | |
| LIB \_05 | A | |  | | --- | | Регистрация читателя |  |  | | --- | |  | | |  | | --- | | Управление читателями |  |  | | --- | |  | | |  | | --- | | Регистрация нового читателя |  |  | | --- | |  | | |  | | --- | | Новый читатель, которого необходимо зарегистрировать |  |  | | --- | |  | | |  | | --- | | Ввести имя, ID, email, телефон читателя. |  |  | | --- | |  | | |  | | --- | | Читатель успешно зарегистрирован. |  |  | | --- | |  | |
| LIB \_06 | A | |  | | --- | | Просмотр карточки читателя |  |  | | --- | |  | | |  | | --- | | Управление читателями |  |  | | --- | |  | | |  | | --- | | Просмотр и создание карточки читателя |  |  | | --- | |  | | |  | | --- | | Читатель существует в системе. |  |  | | --- | |  | | |  | | --- | | Ввести ID читателя для поиска. |  |  | | --- | |  | | |  | | --- | | Отображается информация о читателе и его история книг. |  |  | | --- | |  | |
| LIB \_07 | A | |  | | --- | | Выдача книги |  |  | | --- | |  | | |  | | --- | | Выдача книг |  |  | | --- | |  | | |  | | --- | |  |   Выдача книги читателю | |  | | --- | | Книга доступна для выдачи, и читатель зарегистрирован. |  |  | | --- | |  | | |  | | --- | | Ввести название книги, ID читателя, ID библиотекаря. |  |  | | --- | |  | | |  | | --- | | Книга успешно выдана читателю. |  |  | | --- | |  | |
| LIB \_08 | A | |  | | --- | | Возврат книги |  |  | | --- | |  | | |  | | --- | | Возврат книг |  |  | | --- | |  | | |  | | --- | |  |   Возврат книги читателем | |  | | --- | | Читатель вернул книгу, и она готова к возврату. |  |  | | --- | |  | | |  | | --- | | Ввести название книги и ID читателя для возврата. |  |  | | --- | |  | | |  | | --- | | Книга успешно возвращена, статус обновлен. |  |  | | --- | |  | |
| LIB \_09 | A | |  | | --- | | Просмотр всех книг |  |  | | --- | |  | | |  | | --- | | Просмотр данных |  |  | | --- | |  | | |  | | --- | |  |   Отображение всех книг в библиотеке | |  | | --- | | В библиотеке есть книги. |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Нажать кнопку "9. Просмотреть все книги" на экране. |  |  | | --- | |  | |  |  | | --- | |  | | Отображается полный список книг в консоли. |
| LIB \_10 | A | |  | | --- | | Просмотр всех данных |  |  | | --- | |  | | |  | | --- | | Просмотр данных |  |  | | --- | |  | | |  | | --- | |  | | Вывод всех данных о библиотеке | |  |  | | --- | |  | | |  | | --- | | Система имеет полные данные о библиотеке. |  |  | | --- | |  | | Нажать кнопку "10. Показать все данные" на экране. | |  | | --- | | Все данные о библиотеке отображаются в консоли. |  |  | | --- | |  | |
| LIB \_11 | B | |  | | --- | | Проверка email |  |  | | --- | |  | | |  | | --- | | Валидатор данных |  |  | | --- | |  | | |  | | --- | | Проверка корректности email |  |  | | --- | |  | | |  | | --- | | Введенный email для читателя или библиотекаря. |  |  | | --- | |  | | |  | | --- | | Ввести корректный и некорректный email для проверки. |  |  | | --- | |  | | |  | | --- | | Email проходит валидацию или выводится ошибка. |  |  | | --- | |  | |
| LIB \_12 | B | |  | | --- | | Проверка телефона |  |  | | --- | |  | | |  | | --- | | Валидатор данных |  |  | | --- | |  | | |  | | --- | | Проверка корректности телефона |  |  | | --- | |  | | |  | | --- | | Введенный телефон для читателя или библиотекаря. |  |  | | --- | |  | | |  | | --- | | Ввести корректный и некорректный телефон для проверки. |  |  | | --- | |  | | |  | | --- | | Телефон проходит валидацию или выводится ошибка. |  |  | | --- | |  | |
| LIB \_13 | A | |  | | --- | | Генерация файла карточки |  |  | | --- | |  | | |  | | --- | | Генерация файлов |  |  | | --- | |  | | |  | | --- | | Создание файла с информацией о читателе |  |  | | --- | |  | | |  | | --- | | Читатель и его история взятых книг, личная информация. |  |  | | --- | |  | | |  | | --- | | Нажать кнопку "6. Просмотр карточки читателя" и выбрать ID читателя. |  |  | | --- | |  | | |  | | --- | | Создается и сохраняется файл .docx с карточкой читателя. |  |  | | --- | |  | |
| LIB \_14 | A | |  | | --- | | Загрузка данных |  |  | | --- | |  | | |  | | --- | | Загрузка данных |  |  | | --- | |  | | |  | | --- | | Загрузка данных из текстового файла |  |  | | --- | |  | | |  | | --- | | Файл с данными библиотеки (authors, books, readers, librarians, borrow\_records). |  |  | | --- | |  | | |  | | --- | | Указать путь к файлу с данными. |  |  | | --- | |  | | |  | | --- | | Данные из файла корректно загружаются в систему. |  |  | | --- | |  | |
| LIB \_15 | A | |  | | --- | | Сохранение данных |  |  | | --- | |  | | |  | | --- | | Сохранение данных |  |  | | --- | |  | | |  | | --- | | Сохранение данных в файл |  |  | | --- | |  | | |  | | --- | | Данные о библиотеке, книгах, читателях, библиотекарях и записях о заимствованиях. |  |  | | --- | |  | | |  | | --- | | Нажать кнопку "0. Выход". |  |  | | --- | |  | | Все данные корректно сохраняются в файл "library\_data.txt". |

# *Приложение 2*

*Диаграммы*

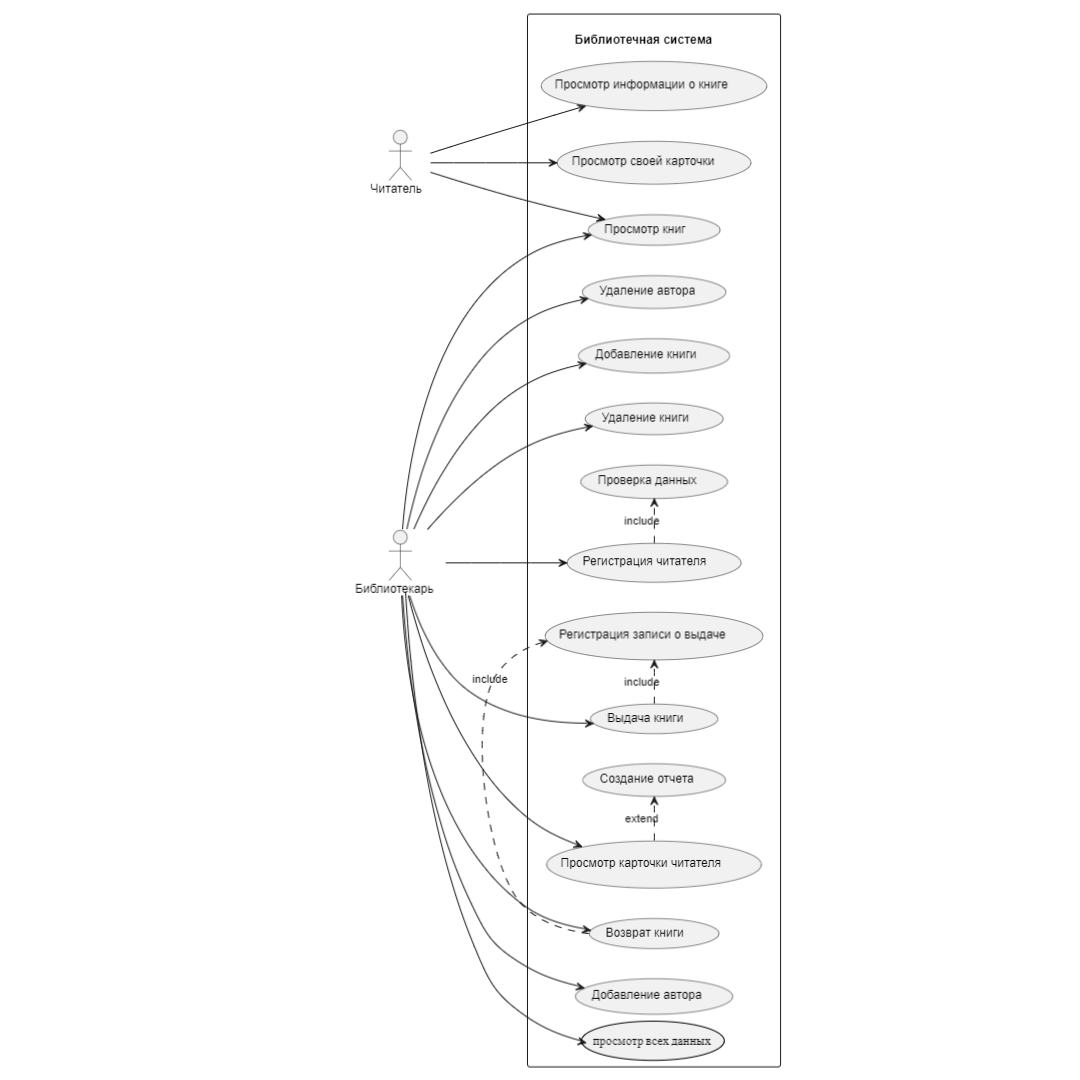


Рисунок 2 – Диаграмма прецендентов

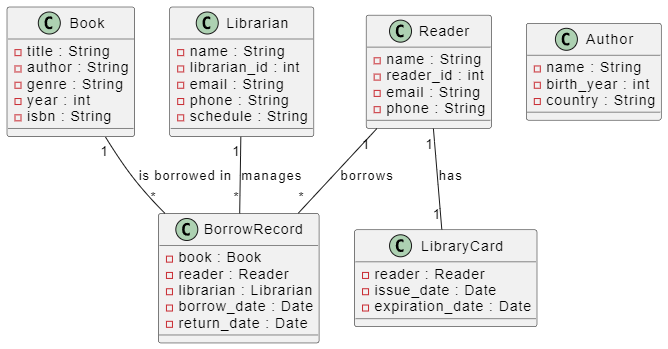


Рисунок 3 – Диаграмма классов

Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

Автоматически созданное описание

Рисунок 4 – Диаграмма IDF1X

# *Приложение 3*

*Полный код программы*

main.py

from file\_handler import load\_data, save\_data

from functions import (

add\_author,

remove\_author,

add\_book,

remove\_book,

register\_reader,

show\_reader\_card,

lend\_book,

return\_book,

show\_all\_books,

show\_all\_data,

)

DATA\_FILE = "data.txt"

def main\_menu():

authors, books, readers, librarians, borrow\_records = load\_data(DATA\_FILE)

while True:

print("-" \* 40)

print("Библиотека")

print("-" \* 40)

print("\nМеню:")

print("1. Добавить автора")

print("2. Удалить автора")

print("3. Добавить книгу")

print("4. Удалить книгу")

print("5. Зарегистрировать читателя")

print("6. Просмотр карточки читателя")

print("7. Выдать книгу")

print("8. Вернуть книгу")

print("9. Просмотреть все книги")

print("10. Показать все данные")

print("0. Выход")

choice = input("Выберите действие: ")

if choice == "1":

add\_author(authors)

elif choice == "2":

remove\_author(authors)

elif choice == "3":

add\_book(books, authors)

elif choice == "4":

remove\_book(books)

elif choice == "5":

register\_reader(readers)

elif choice == "6":

show\_reader\_card(readers, borrow\_records)

elif choice == "7":

lend\_book(books, readers, librarians, borrow\_records)

elif choice == "8":

return\_book(borrow\_records)

elif choice == "9":

show\_all\_books(books)

elif choice == "10":

show\_all\_data(books, authors, readers, librarians, borrow\_records)

elif choice == "0":

save\_data(authors, books, readers, librarians, borrow\_records)

print("Работа завершена. Итоговые данные сохранены.")

break

else:

print("Неверный выбор. Попробуйте снова.")

if \_\_name\_\_ == "\_\_main\_\_":

main\_menu()

file\_handler.py

from models import Author, Book, Reader, Librarian, BorrowRecord

def load\_data(filename):

authors, books, readers, librarians, borrow\_records = [], [], [], [], []

with open(filename, "r", encoding="utf-8") as file:

section = None

for line in file:

line = line.strip()

if line.startswith("[") and line.endswith("]"):

section = line[1:-1]

elif line and section:

if section == "Authors":

name, birth\_year, country = line.split(",")

authors.append(Author(name, int(birth\_year), country))

elif section == "Books":

title, author, genre, year, isbn = line.split(",")

books.append(Book(title, author, genre, int(year), isbn))

elif section == "Readers":

name, reader\_id, email, phone = line.split(",")

readers.append(Reader(name, int(reader\_id), email, phone))

elif section == "Librarians":

name, librarian\_id, email, phone, schedule = line.split(",")

librarians.append(Librarian(name, int(librarian\_id), email, phone, schedule))

elif section == "BorrowRecords":

title, reader\_name, librarian\_name, borrow\_date, return\_date = line.split(",")

book = next((b for b in books if b.title == title), title)

reader = next((r for r in readers if r.name == reader\_name), reader\_name)

librarian = next((l for l in librarians if l.name == librarian\_name), librarian\_name)

borrow\_records.append(BorrowRecord(book, reader, librarian, borrow\_date, return\_date if return\_date != "None" else None))

return authors, books, readers, librarians, borrow\_records

def save\_data(authors, books, readers, librarians, borrow\_records):

filename = "library\_data.txt"

with open(filename, "w", encoding="utf-8") as file:

file.write("[Authors]\n")

for author in authors:

file.write(f"{author.name},{author.birth\_year},{author.country}\n")

file.write("\n[Books]\n")

for book in books:

file.write(f"{book.title},{book.author},{book.genre},{book.year},{book.isbn}\n")

file.write("\n[Readers]\n")

for reader in readers:

file.write(f"{reader.name},{reader.reader\_id},{reader.email},{reader.phone}\n")

file.write("\n[Librarians]\n")

for librarian in librarians:

file.write(

f"{librarian.name},{librarian.librarian\_id},{librarian.email},{librarian.phone},{librarian.schedule}\n")

file.write("\n[BorrowRecords]\n")

for record in borrow\_records:

return\_date = record.return\_date if record.return\_date else "Книга не возвращена"

file.write(f"{record.book.title if isinstance(record.book, Book) else record.book},"

f"{record.reader.name if isinstance(record.reader, Reader) else record.reader},"

f"{record.librarian.name if isinstance(record.librarian, Librarian) else record.librarian},"

f"{record.borrow\_date},{return\_date}\n")

print(f"Итоговый файл сохранен: {filename}")

functions.py

from datetime import datetime

from docx import Document

import re

from models import Author, Book, BorrowRecord, Reader, Librarian

def add\_author(authors):

name = input("Введите имя автора: ")

birth\_year = input("Введите год рождения автора: ")

country = input("Введите страну автора: ")

authors.append(Author(name, int(birth\_year), country))

print(f"Автор {name} добавлен.")

def remove\_author(authors):

name = input("Введите имя автора для удаления: ")

for author in authors:

if author.name.lower() == name.lower():

authors.remove(author)

print(f"Автор {name} удален.")

return

print("Автор не найден.")

def add\_book(books, authors):

title = input("Введите название книги: ")

author\_name = input("Введите имя автора: ")

genre = input("Введите жанр книги: ")

year = input("Введите год издания книги: ")

isbn = input("Введите ISBN книги: ")

author = next((a for a in authors if a.name.lower() == author\_name.lower()), None)

if not author:

print(f"Автор {author\_name} не найден. Сначала добавьте автора.")

return

books.append(Book(title, author.name, genre, int(year), isbn))

print(f"Книга '{title}' добавлена.")

def remove\_book(books):

title = input("Введите название книги для удаления: ")

for book in books:

if book.title.lower() == title.lower():

books.remove(book)

print(f"Книга '{title}' удалена.")

return

print("Книга не найдена.")

def register\_reader(readers):

name = input("Введите имя читателя: ")

reader\_id = len(readers) + 1

email = input("Введите email читателя: ")

phone = input("Введите телефон читателя: ")

readers.append(Reader(name, reader\_id, email, phone))

print(f"Читатель {name} зарегистрирован.")

def show\_reader\_card(readers, borrow\_records):

reader\_id = int(input("Введите ID читателя: "))

reader = next((r for r in readers if r.reader\_id == reader\_id), None)

if reader:

print("\nКарточка читателя:")

print(f" Имя: {reader.name}")

print(f" Email: {reader.email}")

print(f" Телефон: {reader.phone}")

print(f" ID читателя: {reader.reader\_id}")

card\_expiry = datetime(2027, 12, 31)

print(f"Срок действия карты: до {card\_expiry.strftime('%d.%m.%Y')}")

borrowed\_books = [record for record in borrow\_records if record.reader == reader]

if borrowed\_books:

print("\nИстория взятых книг:")

for record in borrowed\_books:

return\_date = record.return\_date if record.return\_date else "Не возвращена"

print(f" Книга: {record.book.title} — Автор: {record.book.author}, Жанр: {record.book.genre}, Год: {record.book.year}")

print(f" Дата выдачи: {record.borrow\_date}")

print(f" Дата возврата: {return\_date}")

print("-" \* 40)

generate\_reader\_card\_file(reader, borrowed\_books)

else:

print("\nУ этого читателя нет взятых книг.")

else:

print("Читатель с таким ID не найден.")

def lend\_book(books, readers, librarians, borrow\_records):

book\_title = input("Введите название книги: ")

book = next((b for b in books if b.title == book\_title), None)

if not book:

print("Книга не найдена.")

return

reader\_id = input("Введите ID читателя: ")

reader = next((r for r in readers if str(r.reader\_id) == reader\_id), None)

if not reader:

print("Читатель с таким ID не найден.")

return

librarian\_id = input("Введите ID библиотекаря: ")

librarian = next((l for l in librarians if str(l.librarian\_id) == librarian\_id), None)

if not librarian:

print("Библиотекарь с таким ID не найден.")

return

for record in borrow\_records:

if record.book.title == book\_title and record.reader.reader\_id == reader.reader\_id and not record.return\_date:

print(f"Ошибка: Книга '{book\_title}' уже выдана читателю с ID {reader.reader\_id}.")

return

borrow\_date = input("Введите дату выдачи книги (ГГГГ-ММ-ДД): ")

borrow\_records.append(BorrowRecord(book, reader, librarian, borrow\_date))

print(f"Книга '{book\_title}' успешно выдана читателю {reader.name}.")

def return\_book(borrow\_records):

book\_title = input("Введите название возвращаемой книги: ")

reader\_id = input("Введите ID читателя: ")

for record in borrow\_records:

if record.book.title == book\_title and str(record.reader.reader\_id) == reader\_id and not record.return\_date:

return\_date = input("Введите дату возврата книги (ГГГГ-ММ-ДД): ")

record.return\_date = return\_date

print(f"Книга '{book\_title}' успешно возвращена читателем {record.reader.name}.")

return

print("Ошибка: Запись о выдаче книги не найдена или книга уже была возвращена.")

def show\_all\_books(books):

if not books:

print("В библиотеке пока нет книг.")

return

print("\nСписок всех книг в библиотеке:")

for i, book in enumerate(books, start=1):

print(f"{i}. {book.title} — Автор: {book.author}, Жанр: {book.genre}, Год: {book.year}, ISBN: {book.isbn}")

def show\_all\_data(books, authors, readers, librarians, borrow\_records):

print("\nАвторы:")

for author in authors:

print(f" {author.name} ({author.birth\_year}) - {author.country}")

print("\nКниги:")

for book in books:

print(f" {book.title} ({book.year}) - {book.author} [ISBN: {book.isbn}]")

print("\nЧитатели:")

for reader in readers:

print(f" {reader.name} (ID: {reader.reader\_id}) - {reader.email}, {reader.phone}")

print("\nБиблиотекари:")

for librarian in librarians:

print(f" {librarian.name} (ID: {librarian.librarian\_id}) - {librarian.email}, {librarian.phone}, {librarian.schedule}")

print("\nЗаписи о выдаче:")

for record in borrow\_records:

book\_title = record.book.title if isinstance(record.book, Book) else record.book

reader\_name = record.reader.name if isinstance(record.reader, Reader) else record.reader

librarian\_name = record.librarian.name if isinstance(record.librarian, Librarian) else record.librarian

status = f"Возвращена: {record.return\_date}" if record.return\_date else "На руках"

print(f" {book\_title} -> {reader\_name}, выдана: {librarian\_name} ({status})")

def validate\_email(email):

pattern = r"^[\w\.-]+@[\w\.-]+\.\w+$"

return re.match(pattern, email)

def validate\_phone(phone):

return phone.isdigit()

def register\_reader(readers):

name = input("Введите имя читателя: ")

reader\_id = input("Введите ID читателя: ")

email = input("Введите email читателя: ")

while not validate\_email(email):

print("Неверный формат email. Попробуйте снова.")

email = input("Введите email читателя: ")

phone = input("Введите номер телефона читателя: ")

while not validate\_phone(phone):

print("Номер телефона должен состоять только из цифр. Попробуйте снова.")

phone = input("Введите номер телефона читателя: ")

readers.append(Reader(name, int(reader\_id), email, phone))

print("Читатель успешно зарегистрирован.")

def generate\_reader\_card\_file(reader, borrowed\_books):

doc = Document()

doc.add\_heading("Карточка читателя", level=1)

doc.add\_paragraph(f"Имя: {reader.name}")

doc.add\_paragraph(f"Email: {reader.email}")

doc.add\_paragraph(f"Телефон: {reader.phone}")

doc.add\_paragraph(f"ID читателя: {reader.reader\_id}")

card\_expiry = datetime(2027, 12, 31)

doc.add\_paragraph(f"Срок действия карты: до {card\_expiry.strftime('%d.%m.%Y')}")

doc.add\_heading("История взятых книг", level=2)

if borrowed\_books:

for record in borrowed\_books:

return\_date = record.return\_date if record.return\_date else "Не возвращена"

doc.add\_paragraph(f"Книга: {record.book.title}")

doc.add\_paragraph(f" Автор: {record.book.author}")

doc.add\_paragraph(f" Жанр: {record.book.genre}")

doc.add\_paragraph(f" Год: {record.book.year}")

doc.add\_paragraph(f" Дата выдачи: {record.borrow\_date}")

doc.add\_paragraph(f" Дата возврата: {return\_date}")

doc.add\_paragraph("-" \* 40)

else:

doc.add\_paragraph("У этого читателя нет взятых книг.")

filename = f"Читатель\_{reader.reader\_id}\_{reader.name.replace(' ', '\_')}.docx"

doc.save(filename)

print(f"Файл карточки читателя сохранен как {filename}.")

models.py

class Book:

def \_\_init\_\_(self, title, author, genre, year, isbn):

self.title = title

self.author = author

self.genre = genre

self.year = year

self.isbn = isbn

class Author:

def \_\_init\_\_(self, name, birth\_year, country):

self.name = name

self.birth\_year = birth\_year

self.country = country

class Reader:

def \_\_init\_\_(self, name, reader\_id, email, phone):

self.name = name

self.reader\_id = reader\_id

self.email = email

self.phone = phone

class LibraryCard:

def \_\_init\_\_(self, reader, issue\_date, expiration\_date):

self.reader = reader

self.issue\_date = issue\_date

self.expiration\_date = expiration\_date

class Librarian:

def \_\_init\_\_(self, name, librarian\_id, email, phone, schedule):

self.name = name

self.librarian\_id = librarian\_id

self.email = email

self.phone = phone

self.schedule = schedule

class BorrowRecord:

def \_\_init\_\_(self, book, reader, librarian, borrow\_date, return\_date=None):

self.book = book

self.reader = reader

self.librarian = librarian

self.borrow\_date = borrow\_date

self.return\_date = return\_date

data.txt

[Authors]

Александр Пушкин,1799,Россия

Лев Толстой,1828,Россия

Фёдор Достоевский,1821,Россия

Джоан Роулинг,1965,Великобритания

Анна Джейн,1987,Россия

Стефани Майер,1973,США

Сара Дж.Маас,1986,США

[Books]

Евгений Онегин,Александр Пушкин,Роман,1833,978-5-699-12011-8

Война и мир,Лев Толстой,Роман,1869,978-5-389-07499-0

Преступление и наказание,Фёдор Достоевский,Роман,1866,978-5-17-087141-6

Гарри Поттер и философский камень,Джоан Роулинг,Фэнтези,1997,978-5-17-118366-2

Гарри Поттер и тайная комната,Джоан Роулинг,Фэнтези,1998,978-5-17-118367-9

Гарри Поттер и узник Азкабана,Джоан Роулинг,Фэнтези,1999,978-5-17-118368-6

Неогранённые алмазы,Анна Джейн,Роман,2016,978-5-699-84376-5

Тысяча сияющих солнц,Анна Джейн,Роман,2017,978-5-699-98865-6

Восхитительная ведьма,Анна Джейн,Роман,2022,978-5-00154-854-6

Сумерки,Стефани Майер,Фэнтези,2005,978-5-17-057028-9

Новолуние,Стефани Майер,Фэнтези,2006,978-5-17-057029-6

Королевство шипов и роз,Сара Дж.Маас,Фэнтези,2015,978-5-17-104972-2

Империя бурь,Сара Дж.Маас,Фэнтези,2016,978-5-17-118442-3

[Readers]

Иван Иванов,1,ivanov@mail.ru,89001234567

Мария Петрова,2,petrova@mail.ru,89007654321

Алексей Смирнов,3,smirnov@mail.ru,89006543210

Елена Соколова,4,sokolova@mail.ru,89007778899

Анастасия Романова,5,romanova@mail.ru,89007557324

[Librarians]

Анна Смирнова,101,anna.smirnova@mail.ru,89005557788,Пн-Пт 9:00-18:00

Олег Кузнецов,102,oleg.kuznetsov@mail.ru,89004446655,Пн-Пт 10:00-19:00

[BorrowRecords]

Гарри Поттер и философский камень,Иван Иванов,Анна Смирнова,2024-11-01,None

Неогранённые алмазы,Мария Петрова,Олег Кузнецов,2024-11-10,None

Сумерки,Алексей Смирнов,Анна Смирнова,2024-11-15,None

Королевство шипов и роз,Елена Соколова,Олег Кузнецов,2024-11-20,None

Восхитительная ведьма,Анастасия Романова,Анна Смирнова,2024-12-12,None