МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И   
ТЕХНОЛОГИЧЕСКОГО ОБРАЗОВАНИЯ**

**Кафедра информационных технологий и электронного обучения**

Основная профессиональная образовательная программа

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль) «Технологии разработки программного обеспечения»

форма обучения – очная

**ДИПЛОМНАЯ РАБОТА**

«Разработка приложения на Python для анализа данных из файлов электронных таблиц»

Обучающейся 4 курса

Иванова Мария Алексеевна

Научный руководитель:

Кандидат физико-математических наук,   
доцент кафедры ИТиЭО

Жуков Николай Николаевич

Санкт-Петербург

2024 г.

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc167292474)

[Глава 1. Анализ предметной области 4](#_Toc167292475)

[1.1 Анализа данных из электронных таблиц 4](#_Toc167292476)

[1.2 Анализ существующих инструментов библиотечной деятельности 5](#_Toc167292477)

[1.3 Требования к инструменту библиотечной деятельности 8](#_Toc167292478)

[1.4 Описание форматов данных электронных таблиц 9](#_Toc167292479)

[1.5 Считывание данных электронных таблиц в python 11](#_Toc167292480)

[1.6 Подготовка данных электронных таблиц в python 14](#_Toc167292481)

[1.7 Анализ данных электронных таблиц 18](#_Toc167292482)

[1.8 Визуализация данных электронных таблиц в python 20](#_Toc167292483)

[Глава 2. Проектирование и разработка приложения 22](#_Toc167292484)

[2.1 Проектирование модели базы данных 22](#_Toc167292485)

[2.2 Описание структуры приложения 24](#_Toc167292486)

[2.3 Модуль пользовательского интерфейса 26](#_Toc167292487)

[2.4 Модуль работы с базой данных SQLite3 33](#_Toc167292488)

[2.5 Модуль импорта данных из XLSX 38](#_Toc167292489)

[2.6 Модуль экспорта данных в XLSX 40](#_Toc167292490)

[2.7 Модуль анализа данных 41](#_Toc167292491)

[ЗАКЛЮЧЕНИЕ 43](#_Toc167292492)

[СПИСОК ЛИТЕРАТУРЫ 44](#_Toc167292493)

[ПРИЛОЖЕНИЕ 45](#_Toc167292494)

ВВЕДЕНИЕ

В современном мире объем данных, с которыми приходится работать различным организациям, постоянно увеличивается. Особенно актуально это для библиотек, где необходимо не только хранить данные о большом количестве книг, но и эффективно управлять информацией о читателях, выданных и возвращенных книгах. Одним из наиболее распространенных форматов хранения данных являются электронные таблицы.

Электронные таблицы стали неотъемлемым инструментом для хранения и обработки данных в различных организациях, в том числе и в школьных библиотеках. Однако, несмотря на их удобство, часто возникает необходимость в разработке специализированных приложений для работы с данными, которые могут предоставлять более мощные и гибкие возможности по сравнению с традиционными электронными таблицами.

Цель данной выпускной квалификационной работы заключается в разработке приложения на языке Python для анализа данных из файлов электронных таблиц, которое будет применяться в школьной библиотеке. Конкретным заказчиком данного проекта выступает МБОУ СОШ 12, которая нуждается в программном обеспечении для учета и управления учебниками. Программа позволит не только считывать и обрабатывать данные, но и сохранять их в базу данных, обеспечивая тем самым надежное хранение и удобный доступ к информации.

Разработка такого приложения позволит автоматизировать множество задач, связанных с управлением библиотечным фондом, что позволит не только повысит точность и скорость обработки данных, но и освободит сотрудников библиотеки от рутинных операций, позволяя им сосредоточиться на более важных аспектах своей работы.

В рамках данной работы поставлены следующие задачи:

1. Анализ существующих инструментов библиотечной деятельности
2. Анализ требований к инструменту библиотечной деятельности
3. Изучить формат электронных таблиц, применяемый в библиотечной деятельности
4. Анализ существующих инструментов чтения форматов электронных таблиц
5. Анализ существующих инструментов хранения данных электронных таблиц
6. Анализ существующих инструментов визуализации электронных таблиц
7. Проектирование и разработка приложения, учитывая требования к инструменту библиотечной деятельности
8. Анализ предметной области
   1. Анализа данных из электронных таблиц

Анализ данных является ключевым процессом в современной информационной эпохе, где объем и разнообразие данных постоянно растут.

В самом общем смысле анализ данных — это исследования, связанные с обсчетом многомерной системы данных, имеющей множество параметров. В процессе анализа данных исследователь производит совокупность действий с целью формирования определенных представлений о характере явления, описываемого этими данными. Как правило, для анализа данных используются различные математические методы [1, 6].

В контексте разработки приложения для анализа данных из файлов электронных таблиц, анализ данных означает извлечение информации из электронных таблиц и преобразование её в удобный для понимания и использования формат.

Этот процесс включает в себя несколько этапов:

* Считывание данных: для анализа необходимо сначала получить данные из электронных таблиц. Это включает в себя чтение информации из файлов формата.
* Подготовка данных: после считывания данных, часто требуется их предварительная обработка. Это может включать в себя очистку данных от ошибок, заполнение пропущенных значений, преобразование форматов и другие манипуляции для обеспечения качественного анализа.
* Анализ данных: здесь происходит основная работа по изучению данных с целью выявления закономерностей, трендов, аномалий и другой информации, которая может быть полезна для принятия управленческих решений.
* Визуализация результатов: для наглядного представления полученных результатов анализа часто используются графики, диаграммы и другие методы визуализации данных. Это позволяет лучше понять характеристики данных и делать выводы на основе их визуального представления.
* Интерпретация и выводы: на последнем этапе анализа данные интерпретируются с целью сформулировать выводы и рекомендации, которые могут быть использованы для улучшения библиотечных процессов или принятия других управленческих решений.

После завершения этапов анализа данных, полученные выводы становятся основой для принятия решений в различных областях.

Например, в контексте школьной библиотеки, анализ данных из электронных таблиц становится важным этапом для эффективного управления информацией о книгах, читателях, выдачах и т. д. Использование Python для анализа и обработки данных из электронных таблиц обеспечит автоматизацию многих рутинных задач и повысит эффективность работы библиотекарей.

* 1. Анализ существующих инструментов библиотечной деятельности

Школьная библиотека играет важную роль в образовательном процессе, предоставляя доступ к различным учебным и художественным материалам. С учетом значимости данного института, анализ существующих инструментов библиотечной деятельности становится важным этапом при разработке приложения для автоматизации учета книг и организации работы библиотеки. В данном разделе будут рассмотрены несколько популярных инструментов, предназначенных для учета книг и управления библиотечными ресурсами.

MyLib[2] — это бесплатная программа для учета книг и управления библиотечными ресурсами, разработанная специально для школьных библиотек.

Учет книг[3] — решение для ведения базы данных печатной продукции, позволяющее вести учёт изданий, содержащихся в библиотеке. Поможет создать развёрнутую картотеку, включающую данные о книгах, журналах, авторах, издательствах, читателях. Данный каталогизатор предназначен для владельцев библиотек и обладателей крупных коллекций книг, библиотекарей и секретарей [4].

Школьная БИБЛИОТЕКА[4] — это еще одна программа для учета книг и управления библиотечными ресурсами, разработанная специально для школьных библиотек.

Рассмотрим указанные раннее существующие инструменты библиотечной деятельности подробнее:

Хранение книг:

* MyLib: осуществляется хранение информации о книгах, включая название, автора, жанр, год издания.
* Учет книг: осуществляется хранение информации о книгах, включая данные об авторе, жанре, издательстве, тираже, серии, формате, годе издания, количестве страниц, ISBN и других характеристиках.
* Школьная БИБЛИОТЕКА: осуществляется хранение информации о книгах, включая информацию об авторе, названии, издательстве и годе издания.

Хранение читателей:

* MyLib: осуществляется хранение информации о читателях, а именно ФИО.
* Учет книг: осуществляется хранение информации о читателей, включая их персональные данные и историю взаимодействия с библиотекой. Каждый читатель имеет уникальный профиль, в котором отражаются выданные на руки издания.
* Школьная БИБЛИОТЕКА: осуществляется хранение информации о читателей, включая информацию о ФИО, классе и контактных данных.

Выдача и возврат книг:

* MyLib: осуществляется функции процессом выдачи и возврата книг читателям, с сохранением сроков выдачи книги.
* Учет книг: осуществляется функции процессом выдачи и возврата книг читателям, с отслеживанием их возврат, а также учитывается сроки и штрафы за просрочку.
* Школьная БИБЛИОТЕКА: осуществляется функции процессом выдачи и возврата книг читателям с автоматическим контролем сроков.

Поиск и фильтрация:

* MyLib: осуществляется поиск книг по различным критериям, таким как автор или название.
* Учет книг: система предоставляет инструменты для удобного поиска и фильтрации книг по различным критериям, таким как автор, жанр, ключевые слова, издательство и другие характеристики.
* Школьная БИБЛИОТЕКА: осуществляется поиск и фильтрация книг по различным параметрам, таким как предметная область и учебный предмет.

Экспорт и импорт данных:

* MyLib: отсутствует функция экспорта и импорта данных из программы. Для переноса данных между программами необходимо выполнять процесс вручную через сторонние приложения, такие как MS Access или OO Base.
* Учет книг: осуществляется экспортирование данных в форматы Excel, Word и CSV.
* Школьная БИБЛИОТЕКА: осуществляется импортирование списков учителей, учеников и классов из таблиц Excel, обеспечивая быструю настройку программы.

Отчетность и статистика:

* MyLib: отсутствует функции отчетности и статистики.
* Учет книг: осуществляется разнообразные отчеты о состоянии библиотечного фонда, активности читателей и других аспектах работы библиотеки.
* Школьная БИБЛИОТЕКА: осуществляется разнообразные отчеты, включая книгу суммарного учета, потребность, задолженности и другие. Возможность сохранения отчетных форм в привычном Excel формате для удобного обмена данными и работы с ними.

В результате сравнительного анализа существующих инструментов библиотечной деятельности для школьных библиотек становится очевидным, что каждый из них обладает своими преимуществами и ограничениями. MyLib предоставляет базовый функционал для учета книг, в то время как Учет книг предлагает более широкие возможности по управлению данными и отчетности. Школьная БИБЛИОТЕКА, в свою очередь, ориентирована на конкретные потребности школьной среды и предоставляет наиболее полный спектр функций.

В целом, анализ существующих инструментов библиотечной деятельности для школьной библиотеки показывает, что существующие решения предоставляют базовый функционал для учета книг и организации работы библиотеки.

Однако, многие существующие информационные системы сложны в использовании из-за обилия функций, которые не всегда необходимы конкретной библиотеке. Также существующие информационные системы имеют высокую стоимость лицензий, что может быть нецелесообразно для небольших библиотек, включая школьные. Таким образом, разработка специализированного приложения на Python для анализа данных из файлов электронных таблиц и управления школьной библиотекой представляется актуальной и перспективной задачей.

* 1. Требования к инструменту библиотечной деятельности

На основе сравнительного анализа существующих инструментов библиотечной деятельности для школьных библиотек можно выделить ряд ключевых требований к разрабатываемому инструменту. Эти требования определяют функциональность, удобство использования и адаптацию к конкретным потребностям школьной библиотеки.

Хранение данных о книгах: инструмент должен обеспечивать возможность хранения разнообразной информации о книгах, включая название, автора, издательство, год издания. Это позволит библиотекарям эффективно управлять библиотечным фондом и обеспечивать доступ к необходимым материалам.

Хранение данных о читателях: инструмент должен предоставлять возможность вести учет информации о читателях, включая ФИО, класс. Это обеспечит эффективное управление выдачей и возвратом книг.

Учет выдачи и возврата книг: инструмент должен поддерживать функции выдачи и возврата книг, обеспечивая контроль за сроками выдачи, учетом просрочек.

Поиск и фильтрация: инструмент должен обеспечивать возможность быстрого и удобного поиска книг по различным критериям, чтобы пользователи могли быстро находить необходимые материалы

Экспорт и импорт данных: инструмент должен поддерживать функции экспорта и импорта данных, позволяя обменять информацией с другими приложениями и системами без потери целостности данных. Это обеспечит удобство взаимодействия с внешними ресурсами и программным обеспечением.

Отчетность и статистика: инструмент должен предоставлять возможность генерации разнообразных отчетов. Это поможет библиотекарям принимать информированные решения и оптимизировать библиотечные процессы.

Простой и интуитивно понятный интерфейс: инструмент должен иметь простой и понятный интерфейс, который позволит пользователям легко освоить его функционал и эффективно использовать приложение без дополнительного обучения.

Для реализации всех вышеуказанных требований, необходимо использовать надежные и проверенные форматы данных, которые обеспечат удобство и эффективность работы с информацией. Одним из таких форматов является XLSX, который широко используется для организации, хранения и анализа данных.

* 1. Описание форматов данных электронных таблиц

Электронные таблицы являются одним из самых распространенных средств для организации, хранения и анализа данных. Они широко используются в различных областях, включая образовательные учреждения, для хранения, обработки и обмена данными. В этой главе мы рассмотрим историю появления формата данных электронных таблиц XLSX, который широко используется в настоящее время.

История появления формата XLSX связана с эволюцией технологий и потребностями пользователей. В начале своего пути Microsoft Excel использовал формат XLS, основанный на бинарной структуре данных.

Формат XLS

Формат XLS был введен компанией Microsoft и использовался в первых версиях Microsoft Excel до 2007 года. Он основан на бинарной структуре данных и поддерживает разнообразные функции, такие как:

* Формулы и функции: в XLS можно сохранять сложные вычислительные формулы и встроенные функции Excel.
* Форматирование: поддержка различных стилей ячеек, шрифтов, цветов, границ и других элементов форматирования.
* Графики и диаграммы: включение различных типов графиков и диаграмм для визуализации данных.
* Макросы: возможность хранения и выполнения макросов на языке VBA (Visual Basic for Applications).

Однако бинарная природа XLS делает его менее гибким и сложным для обработки и анализа с использованием современных программных средств, таких как Python. Кроме того, формат XLS имеет ограничения по объему данных, что может стать проблемой при работе с большими таблицами.

**Формат XLSX**

С 2007 года Microsoft Excel перешел на новый формат — XLSX, который основан на открытых стандартах и использует XML для хранения данных.

Формат XML (Extensible Markup Language) — это текстовый формат, предназначенный для описания структур данных. При этом данные заключаются в специальные теги, которые предназначены для описания данных[6, 648].

Основные особенности формата XLSX по сравнению с XLS включают:

* Модульная структура: файл XLSX представляет собой ZIP-архив, содержащий несколько XML-файлов, каждый из которых отвечает за определенную часть данных (сами данные, стили, графики и т.д.).
* Совместимость и расширяемость: благодаря использованию XML, формат XLSX легко интегрируется с другими системами и может быть расширен для специфических нужд.
* Увеличенные объемы данных: формат XLSX поддерживает более крупные таблицы по сравнению с XLS, что позволяет работать с большими объемами данных.
* Улучшенная производительность: более эффективная структура хранения данных позволяет быстрее загружать и обрабатывать большие файлы.

Для более наглядного понимания различий между форматами XLS и XLSX, приведем таблицу:

|  |  |  |
| --- | --- | --- |
| Характеристика | XLS | XLSX |
| Год выпуска | До 2007 года | С 2007 года |
| Структура данных | Бинарная | Основана на XML |
| Размер файла | Обычно крупнее | Меньше благодаря формату XML и сжатию данных |
| Максимальный объем данных | 65 536 строк и 256 столбцов | 1 048 576 строк и 16 384 столбцов |
| Форматирование | Поддержка старых функции Excel | Поддержка различных стилей, шрифтов и цветов |
| Графики и диаграммы | Ограничен старыми типами диаграмм Excel | Поддержка новых функции и типы диаграмм Excel |
| Макросы | Поддержка VBA | Не поддерживает макросы |
| Совместимость | Ограниченная совместимость с новыми системами | Высокая совместимость с современными системами и приложениями |

На основе информации из таблицы о форматах электронных таблиц (XLS и XLSX) можно сделать вывод, что формат XLSX обладает рядом преимуществ по сравнению с форматом XLS.

Формат XLSX лучше интегрируется с современными языками программирования и библиотеками, такими как Python. Библиотеки openpyxl и pandas предоставляют мощные инструменты для чтения, записи и анализа данных в формате XLSX. Это упрощает разработку приложения и позволяет использовать современные методологии программирования.

Также в отличие от XLS, формат XLSX поддерживает большие объемы данных, что позволяет работать с обширными таблицами.

Исходя из вышеуказанных причины, эти изменения сделали формат XLSX предпочтительным выбором для многих пользователей и разработчиков, и он стал стандартом для многих приложений, включая различные программные решения для анализа данных, такие как приложение для школьной библиотеки, описанное в данной работе.

В следующей главе будет рассмотрено, как именно происходит считывание данных из файлов XLSX в среде Python и подготовка данных для дальнейшей обработки и анализа, что будет являться ключевым этапом в разработке приложения для школьной библиотеки.

* 1. Считывание данных электронных таблиц в python

Для работы с данными из электронных таблиц в формате XLSX важно выбрать подходящую библиотеку для работы с этим типом файлов. В Python существует несколько библиотек, которые предоставляют удобные инструменты для чтения, записи и анализа данных из электронных таблиц данного формата.

В этом разделе рассмотрим основные библиотеки Python, которые позволяют эффективно работать с данными из электронных таблиц: openpyxl, xlrd и pandas.

**openpyxl** — библиотека Python для чтения/записи файлов Excel 2010[7]. Она позволяет работать с файлами Excel на высоком уровне, предоставляя инструменты для создания и изменения сложных таблиц, графиков и других элементов.

Основные возможности openpyxl

* Чтение и запись файлов: openpyxl позволяет открывать существующие файлы Excel, читать данные из них, а также создавать новые файлы и записывать данные.
* Работа с ячейками и диапазонами: библиотека предоставляет доступ к отдельным ячейкам и диапазонам, что позволяет гибко манипулировать данными.
* Форматирование: возможность изменять стили ячеек, применять форматирование, добавлять стили и шрифты.
* Создание и модификация диаграмм: openpyxl поддерживает создание различных типов диаграмм, включая гистограммы, линейные диаграммы и диаграммы рассеяния.
* Поддержка формул: возможность чтения и записи формул в ячейках.
* Обработка защищенных паролем файлов: поддержка чтения и записи файлов Excel, защищенных паролем.

**xlrd** — библиотека для чтения данных и форматирования информации из Excel файлов[8]. Библиотека поддерживает форматы XLS и частично XLSX. Она была одной из первых библиотек для работы с Excel в Python и широко используется для извлечения данных из старых форматов файлов.

Основные возможности xlrd

* Чтение файлов XLS и XLSX: xlrd позволяет открывать и читать данные из файлов Excel форматов XLS и частично XLSX.
* Извлечение данных: доступ к данным ячеек, включая формулы и форматирование.
* Обработка листов: возможность работать с несколькими листами в одном файле, извлекая данные из каждого листа.
* Поддержка базовых форматов данных: чтение основных типов данных, включая текст, числа и даты.

Библиотека **pandas** предоставляет структуры данных и функции, призванные сделать работу со структурированными данными. простым, быстрым и выразительным делом [9, 25]. В контексте работы с Excel, pandas предоставляет удобный интерфейс для чтения и записи данных в формате XLSX.

Основные возможности pandas

* Чтение и запись файлов Excel: pandas позволяет легко читать и записывать данные в файлы Excel с помощью функций read\_excel и to\_excel.
* Мощные структуры данных: использование DataFrame и Series для организации данных, что упрощает анализ и манипуляции.
* Интеграция с другими инструментами анализа данных: pandas хорошо интегрируется с библиотеками numpy и matplotlib, что позволяет проводить сложный анализ и визуализацию данных.
* Гибкие возможности фильтрации и агрегации данных: мощные функции для фильтрации, сортировки, группировки и агрегирования данных.
* Обработка больших объемов данных: pandas оптимизирован для работы с большими объемами данных, обеспечивая высокую производительность.

При выборе библиотеки для работы с данными Excel важно учитывать конкретные задачи и требования проекта.

Ниже приведен сравнительный анализ openpyxl, xlrd и pandas:

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | openpyxl | xlrd | pandas |
| Чтение форматов | XLSX | XLS, частично XLSX | XLSX, XLS |
| Запись форматов | XLSX | Нет | XLSX |
| Форматирование | Да | Ограниченное | Ограниченное |
| Поддержка диаграмм | Да | Нет | Нет |
| Работа с большими данными | Средняя | Высокая | Высокая |
| Интеграция с другими библиотеками | Ограниченная | Ограниченная | Высокая |

Openpyxl является отличным выбором, если требуется полная поддержка форматов Excel и работа с диаграммами и сложным форматированием. xlrd подходит для чтения данных из старых форматов файлов и может быть полезен для задач, где запись данных не требуется. pandas — лучший выбор для анализа данных и интеграции с другими инструментами Python, особенно при обработке больших объемов данных и необходимости в мощных функциях фильтрации и агрегации.

В итоге, pandas выделяется как наилучший выбор для библиотеки при работе с файлами Excel в Python в рамках данного проекта. Эта библиотека обеспечивает не только возможности чтения и записи данных, но и интегрируется эффективно с другими инструментами анализа данных в Python. Кроме того, по сравнению с openpyxl, она не перегружена излишним функционалом по форматированию, что делает её более удобной и эффективной в использовании.

* 1. Подготовка данных электронных таблиц в python

Подготовка данных из электронных таблиц играет ключевую роль в процессе анализа данных, обеспечивая их качественность и пригодность для дальнейших исследований и аналитики. Один из эффективных способов обработки и хранения данных, включая данные из электронных таблиц, — это использование систем управления базами данных (СУБД).

Система управления базами данных (СУБД) — специализированное программное обеспечение, обеспечивающее доступ к базе данных как к совокупности её структурных единиц[10, 9].

СУБД представляют собой программное обеспечение, специально разработанное для управления большими объемами данных, обеспечивая их эффективное хранение, организацию, доступ и анализ. В контексте подготовки данных из электронных таблиц использование СУБД обладает рядом преимуществ:

* Структурированное хранение данных: СУБД позволяют структурированно хранить данные, что упрощает доступ к ним и обработку. Это особенно важно при работе с большими объемами информации из электронных таблиц.
* Масштабируемость: СУБД способны обрабатывать огромные объемы данных, что делает их подходящим выбором для проектов, требующих обработки больших данных из электронных таблиц.
* Безопасность данных: СУБД обеспечивают механизмы защиты данных, включая управление доступом, шифрование и резервное копирование, что помогает предотвратить утечку информации и обеспечить их целостность.
* Язык запросов: большинство СУБД поддерживают языки запросов, такие как SQL, что упрощает выполнение различных операций с данными, включая их извлечение, фильтрацию, сортировку и объединение.

Использование СУБД в процессе подготовки данных из электронных таблиц обеспечивает более эффективную, безопасную и структурированную обработку информации, что в свою очередь способствует более качественному анализу данных и принятию обоснованных управленческих решений.

При выборе подходящей СУБД для конкретного проекта необходимо учитывать требования к производительности, масштабируемости, надежности, а также удобство использования и доступность инструментов для работы с ними.

Ниже приведен обзор основных СУБД, которые могут быть использованы при разработке приложения на Python для анализа данных из файлов электронных таблиц и их сохранения в базу данных:

**SQLite** — это встроенная библиотека, которая часто используется для управления данными в файлах. Будучи файловой базой данных, она предлагает простые и быстрые инструменты для работы с данными, что делает её удобной для небольших приложений.

Преимущества:

* Файловая структура: вся база данных хранится в одном файле, что облегчает её перенос и управление.
* Стандартизированность: использует SQL с небольшими отклонениями, такими как отсутствие RIGHT OUTER JOIN.
* Удобство для разработки и тестирования: простота использования и богатый набор функций делают её отличным инструментом для разработки и тестирования.

Недостатки:

* Отсутствие пользовательского управления: нет возможности управлять правами доступа на уровне пользователя.
* Ограниченная настройка: производительность не может быть улучшена путём настройки

**MySQL** — это одна из самых популярных серверных баз данных. Она широко используется благодаря своей простоте и множеству доступных ресурсов и инструментов.

Преимущества:

* Простота: легко устанавливается и имеет много сторонних инструментов, упрощающих работу.
* Функциональность: поддерживает большинство функций SQL.
* Безопасность: Встроенные функции безопасности.
* Мощность и масштабируемость: может обрабатывать большие объемы данных и поддерживает масштабируемые приложения.
* Скорость: производительность обеспечивается за счёт некоторых отклонений от стандартов.

Недостатки:

* Ограничения функциональности: не все возможности SQL реализованы.
* Надежность: некоторые операции могут быть менее надёжными по сравнению с другими СУБД.

**PostgreSQL** — это мощная, объектно-реляционная база данных, которая полностью соответствует стандартам SQL и предлагает широкий набор расширяемых функций.

Преимущества:

* Полная совместимость с SQL: соответствует стандартам ANSI/ISO.
* Поддержка сторонними инструментами: широкая интеграция с различными инструментами.
* Расширяемость: возможность добавления хранимых процедур и других расширений.
* Объектно-ориентированность: поддерживает объектно-ориентированные принципы наряду с реляционными.

Недостатки:

* Производительность: может уступать конкурентам в простых операциях чтения.
* Хостинг: найти подходящего провайдера может быть сложно из-за его специфичности.

Для наглядного сравнения особенностей и возможностей SQLite, MySQL и PostgreSQL, представим их основные характеристики в табличной форме. Это позволит более четко увидеть различия и преимущества каждой из этих СУБД.

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | SQLite | MySQL | PostgreSQL |
| Тип системы | Встраиваемая, файловая БД | Серверная БД | Серверная БД |
| Формат хранения | Один файл | Таблицы в базе данных, управляемые сервером | Таблицы в базе данных, управляемые сервером |
| SQL-совместимость | Стандартизированная, но некоторые функции отсутствуют | Широкая поддержка SQL, но с некоторыми упрощениями | Полная совместимость с SQL (ANSI/ISO) |
| Управление пользователями | Нет поддержки пользовательского управления | Многоуровневая система управления доступом | Поддержка пользовательского управления |
| Производительность | Высокая для небольших и средних объемов данных | Высокая производительность, оптимизация за счет упрощений | Высокая производительность в многопользовательских сценариях |
| Масштабируемость | Ограничена размером файла | Высокая, поддержка больших объемов данных | Высокая, особенно в многопользовательских системах |
| Безопасность | Базовая | Встроенные функции безопасности | Расширенные функции безопасности |
| Расширяемость | Ограниченная | Сторонние инструменты и плагины | Высокая, возможность добавления хранимых процедур |
| Простота использования | Очень простая установка и использование | Легкая установка и наличие множества сторонних инструментов | Более сложная установка и настройка |
| Объектно-ориентированность | Нет | Нет | Полная поддержка объектно-ориентированного подхода |
| Проблемы с хостингом | Нет | Нет | Возможны проблемы из-за сложности и требований |

Для приложения школьной библиотеки, основная задача которого заключается в простом и эффективном управлении данными, наиболее подходящей СУБД является SQLite3. Её простота, лёгкость и встроенная поддержка в Python делают её идеальным выбором для небольших и несложных приложений. SQLite3 не требует дополнительных настроек и ресурсов.

В случае необходимости расширения функциональности и увеличения нагрузки в будущем, можно рассмотреть переход на более мощные СУБД, такие как PostgreSQL или MySQL. PostgreSQL будет предпочтительнее для приложений, требующих сложных операций с данными и высокой надёжности, тогда как MySQL может быть оптимальным выбором для веб-приложений с большим количеством чтений.

Таким образом, начальное использование SQLite3 является обоснованным выбором, учитывая текущие требования и ресурсы школьной библиотеки.

* 1. Анализ данных электронных таблиц

После того как данные из электронных таблиц собраны и подготовлены для анализа, начинается их обработка и исследование. На этом этапе важно организовать данные, изучить их структуру и применить статистические методы для получения значимой информации. При анализе данных используются различные методы, одним из которых является дескриптивный анализ [11, 88]. Эти методы позволяют обобщить информацию о данных и выделить основные характеристики.

**Распределение частот.**

Распределение частот отображает, как часто встречаются различные значения в наборе данных. Это позволяет выявить, какие значения наиболее распространены, а какие — редки. Частотное распределение может быть представлено в виде таблиц, гистограмм или других графических форм.

**Доли, проценты и пропорции.**

Дескриптивный анализ может включать в себя анализ долей, процентов и пропорций. После построения распределения частот выбирается один из трех типов анализа в зависимости от конкретных задач и целей исследования:

Доли показывают относительную частоту ответов в каждой категории и вычисляются как отношение числа ответов в конкретной категории к общему числу ответов. Это помогает понять, какую часть от общего количества составляют ответы в каждой категории.

Где f - частота, или количество случаев в любой категории;

N - количество случаев во всех категориях.

Проценты представляют собой доли, умноженные на 100. Они позволяют наглядно представить, какой процент от общего числа ответов приходится на каждую категорию.

Где f - частота, или количество случаев в любой категории;

N - количество случаев во всех категориях.

Пропорция — это числовая характеристика, используемая для сравнения количества случаев в различных категориях переменной. Математически она определяется как отношение числа случаев в одной категории к числу случаев в другой категории

Где: f₁ - количество случаев в первой категории;

f₂ - количество случаев во второй категории.

Использование дескриптивного анализ позволяет эффективно анализировать данные, полученные из электронных таблиц, и представлять результаты в наглядной и понятной форме. Это особенно важно для разработки приложений, таких как программа для школьной библиотеки, где результаты анализа помогают в принятии решений и улучшении работы библиотеки.

* 1. Визуализация данных электронных таблиц в python

Визуализация данных является ключевым элементом анализа данных, поскольку она позволяет пользователям быстро и легко понять сложные наборы данных. Для школьной библиотеки это особенно важно, так как визуализация помогает библиотекарям статистику по книговыдачам.

В данной главе рассмотрены основные подходы и инструменты для визуализации данных из электронных таблиц в Python. Особое внимание уделено использованию библиотек для создания графического интерфейса пользователя (GUI). Для создания графического интерфейса пользователя (GUI) в Python существует множество библиотек графического интерфейса, хотя ни одна из них не получила такого широкого распространения, как Tkinter и PyQt.

Tkinter — это модуль Python, который работает с библиотекой Tk. С помощью этой библиотеки написано достаточно большое количество проектов. Основная причина использования этой библиотеки при создании приложений с GUI на языке Python заключается втом, что установочный файл Python уже включает модуль tkinter в состав стандартной библиотеки [12, 7].

PyQt — это среда графического пользовательского интерфейса Python, построенная на основе платформы C++ Qt, которая разрабатывается и поддерживается компанией Qt [13].

В контексте создания инструмента для школьной библиотеки выбор был сделан в пользу библиотеки PyQt. Этот выбор обусловлен несколькими причинами. Во-первых, PyQt предлагает широкий спектр готовых компонентов и инструментов, что значительно упрощает разработку сложных приложений. Это включает в себя средства для работы с базами данных, построения графиков, многопоточности и сетевого взаимодействия. Такие возможности делают PyQt идеальным выбором для создания профессиональных и функционально насыщенных приложений.

PyQt также отличается удобством и простотой разработки. Использование готовых компонентов позволяет минимизировать количество кода, необходимого для создания интерфейса. Например, для отображения данных из файла XLSX можно использовать встроенные табличные представления, которые легко интегрируются с моделью данных. В отличие от PyQt, Tkinter требует больше ручной работы и не предоставляет такого же уровня интеграции и функциональности, что делает его менее подходящим для сложных проектов.

Производительность является еще одним важным аспектом. PyQt эффективен в обработке больших объемов данных. Табличные представления в PyQt могут обрабатывать миллионы строк без значительных потерь производительности, что особенно важно для библиотечной системы, где объем данных может быть значительным. Tkinter, напротив, может испытывать проблемы с производительностью при обработке больших данных, поскольку каждая ячейка в таблице является отдельным объектом виджета, что увеличивает нагрузку на систему.

В заключение, выбор библиотеки для создания графического интерфейса пользователя (GUI) важен для успешной разработки приложений, особенно когда речь идет о приложениях для обработки и визуализации данных, таких как система управления библиотекой. В контексте данного проекта выбор библиотеки PyQt обоснован его мощными возможностями, широким набором компонентов и инструментов, а также высокой производительностью при обработке больших объемов данных. Использование PyQt позволяет создавать профессиональные и функционально насыщенные приложения с минимальными затратами времени и усилий на разработку интерфейса. Таким образом, PyQt является идеальным выбором для создания инструмента управления школьной библиотекой, обеспечивая удобство, производительность и высокий уровень функциональности.

1. Проектирование и разработка приложения
   1. Проектирование модели базы данных

Проектирование модели базы данных является ключевым этапом в разработке приложения для анализа данных из электронных таблиц. В данном разделе будет описана логическая модель базы данных, используемой для хранения информации школьной библиотеки.

Основными компонентами моделирования данных являются:

* Сущность: это различимый и заметный объект, информация о котором должна быть хранится в базе данных;
* Атрибут: представляет характеристики именованного объекта. Каждый атрибут должен быть уникальным для определенного вида объекта, но может быть общим для разных видов объектов;
* Связь: представляет собой связь или отношение между двумя или более объектами. Она указывает на взаимосвязь и зависимость между объектами.

Логическая модель данных детализирует концептуальную модель, добавляя конкретные атрибуты для каждой сущности и определяя типы данных для каждого атрибута. Логическая модель данных не учитывает специфику хранения данных в конкретной СУБД, но уже отражает структуру таблиц и их атрибуты.

Для школьной библиотеки логическая модель данных включает основные сущности:

* Группы (Groups): информация о учебных группах студентов.
* Книги (Books): информация о книгах в библиотеке.
* Дисциплины (Disciplines): информация о учебных дисциплины в школе.
* Студенты (Students): информация о студентах.
* Библиотека (Library): информацию о записи взятых книгах студентами по тому или иному предмету.

Подробное описание полей и ограничений каждой таблицы приведено в приложении A (Таблица 1), что позволяет четко понимать структуру данных и их взаимосвязи.

Связи между сущностями базы данных определяют, как таблицы взаимодействуют друг с другом. Эти связи обеспечивают целостность данных в приложении, а также позволяют эффективно управлять информацией о книгах, студентах и дисциплинах.

На диаграмме (Рисунок 1) представлены связи между таблицами groups, books, disciplines, students и library.

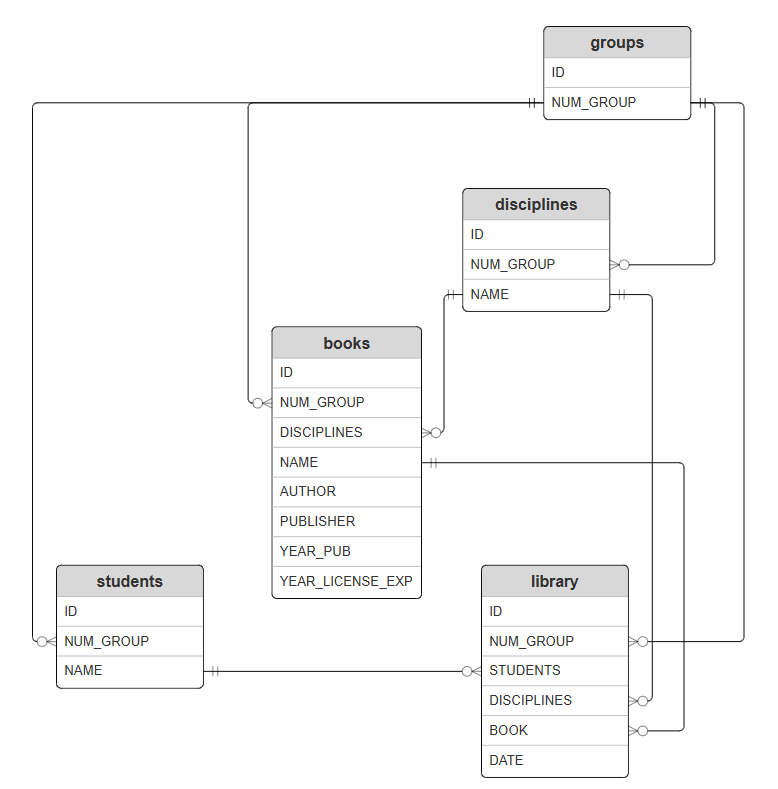


Рисунок 1- Логическая модель данных.

Как видно на диаграмме, представленной в приложении, таблица groups имеет несколько связей "один ко многим". Эти связи существуют с таблицами books, students, library и disciplines, что указывает на то, что одна группа может включать множество книг, студентов, записей библиотеки и дисциплин. Это обеспечивает возможность группировать книги и студентов по различным группам, что важно для учета и организации в школьной библиотеке.

Таблица books связана с таблицей library, что означает, что одна книга может быть связана с множеством записей в библиотеке. Это важно для учета того, как часто и кем использовалась та или иная книга.

Таблица students также связана с таблицей library, что указывает на то, что один студент может иметь множество записей в библиотеке. Это позволяет отслеживать, какие книги были взяты конкретным студентом и когда они были возвращены.

Таблица disciplines имеет связь "один ко многим" с таблицами library и books, что позволяет одной дисциплине быть связанной с множеством записей библиотеки и книг. Это важно для учета того, какие книги используются в рамках той или иной дисциплины.

Эти связи позволяют структурировать данные таким образом, чтобы легко было выполнять различные операции, такие как поиск, добавление, обновление и удаление записей, а также генерацию отчетов. Теперь, когда определены основные сущности и их взаимосвязи в концептуальной модели данных для школьной библиотеки, можно перейти к следующему этапу - описание структуры приложения.

* 1. Описание структуры приложения

Приложение для анализа данных школьной библиотеки состоит из нескольких модулей, каждый из которых выполняет специализированные функции для обеспечения полной функциональности системы. В данной главе рассматривается структура приложения, включая описание функциональности каждого модуля и их взаимодействие.

Код программного обеспечения (ПО) можно просмотреть на GitHub. Для перехода используйте QR-код, представленный на Рисунке 1 (см. Приложение A).

**Модуль пользовательского интерфейса**

Модуль пользовательского интерфейса обеспечивает взаимодействие пользователя с приложением. Этот модуль реализован с использованием различных библиотеки PyQt, и выполняет следующие функции:

* Предоставление интерфейса для работы с данными.
* Инициирование процессов импорта и экспорта данных.
* Отправка запросов на анализ данных и отображение результатов анализа.

**Модуль работы с базой данных SQLite3**

Этот модуль является центральным элементом системы, обеспечивающим взаимодействие с базой данных SQLite3. Основные функции модуля включают:

* Создание и управление таблицами базы данных.
* Создание, обновление, удаление и получение данных из базы данных.

**Модуль импорта данных из XLSX**

Модуль импорта данных отвечает за считывание информации из файлов электронных таблиц формата XLSX и передачу этих данных в базу данных SQLite3. Этот модуль использует библиотеку pandas для работы с XLSX-файлами. Основные функции модуля включают:

* Открытие и чтение данных из XLSX-файла.
* Обработка данных для обеспечения их соответствия требованиям системы.
* Передача обработанных данных в модуль работы с базой данных для хранения в базе данных SQLite3.

**Модуль экспорта данных в XLSX**

Модуль экспорта данных обеспечивает возможность сохранения данных из базы данных SQLite3 в формате XLSX. Этот модуль также использует библиотеку pandas и выполняет следующие задачи:

* Извлечение данных из базы данных SQLite3.
* Формирование XLSX-файла с данными.
* Сохранение и предоставление XLSX-файла пользователю.

**Модуль анализа данных**

Модуль анализа данных предназначен для обработки и анализа данных, хранящихся в базе данных. Основные задачи модуля:

* Запрос данных из базы данных для анализа.
* Обработка и анализ данных в соответствии с запросами пользователя.
* Передача результатов анализа в модуль пользовательского интерфейса.

**Взаимодействие модулей**

Взаимодействие между модулями приложения обеспечивается через четко определенные интерфейсы и последовательность операций. Ниже представлена диаграмма на рисунке 2, иллюстрирующая связи и взаимодействие между модулями.

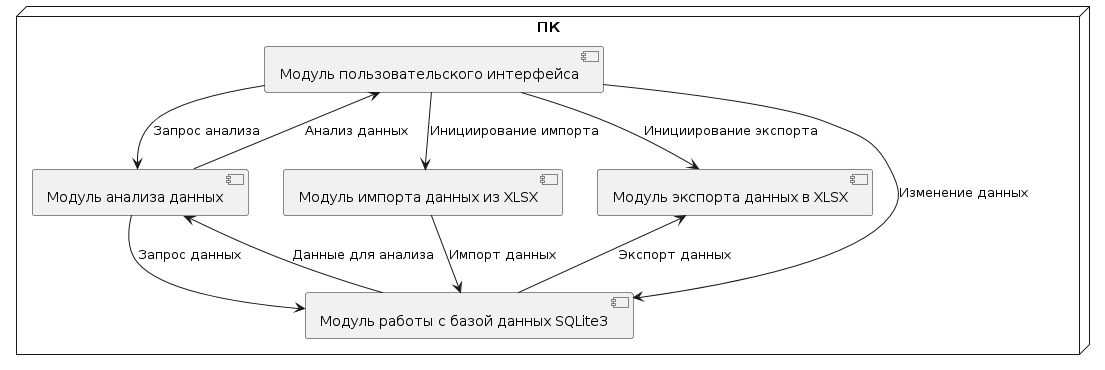


Рисунок 2 - Диаграмма развертывания

Структура приложения, представленная в данной главе, обеспечивает эффективное разделение функциональных обязанностей между модулями, что способствует более легкому развитию и поддержке приложения. Каждый модуль выполняет свои специфические задачи и взаимодействует с другими модулями через четко определенные интерфейсы, что делает систему гибкой и масштабируемой.

* 1. Модуль пользовательского интерфейса

В данной главе рассматривается процесс разработки модуля пользовательского интерфейса приложения для школьной библиотеки, который разработан с использованием библиотеки PyQt5. Этот модуль обеспечивает взаимодействие пользователя с приложением, предоставляя удобные и интуитивно понятные формы для выполнения основных операций, таких как учет книг, добавление групп, дисциплин, учеников и книг.

В данном разделе рассматриваются основные аспекты проектирования и реализации модуля пользовательского интерфейса, используемого в приложении.

**Описание классов интерфейса**

Класс **Ui\_Main** представляет главное окно приложения представление на рисунке 3. Этот класс предоставляет пользователю интерфейс для навигации между основными модулями приложения.

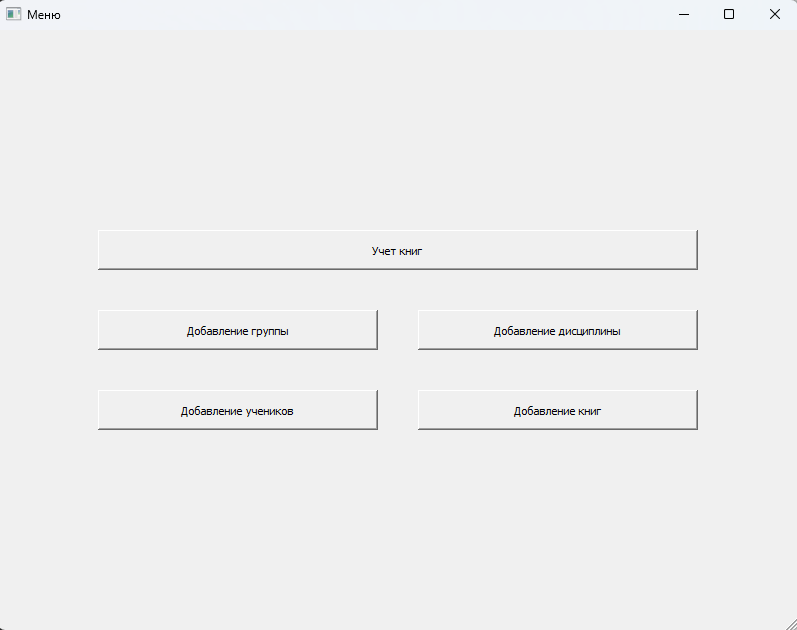


Рисунок 3 - Окно Ui\_Main

В классе инициализируются элементы интерфейса, такие как кнопки для учета книг, добавления группы, дисциплины, учеников и книг. Эти кнопки отвечают за открытие окон, вызывая функции, такие как open\_groups\_window, open\_discipline\_window, open\_student\_window и open\_books\_window.

Каждый из этих функций вызывает экземпляр соответствующего класса окна и отображает это окно, скрывает текущее окно.

Класс **Ui\_library** отвечает за окно “Учет книг” представленное на рисунке 4. Этот класс предоставляет интерфейс для просмотра и редактирования данных о выданных и возвращенных книгах.

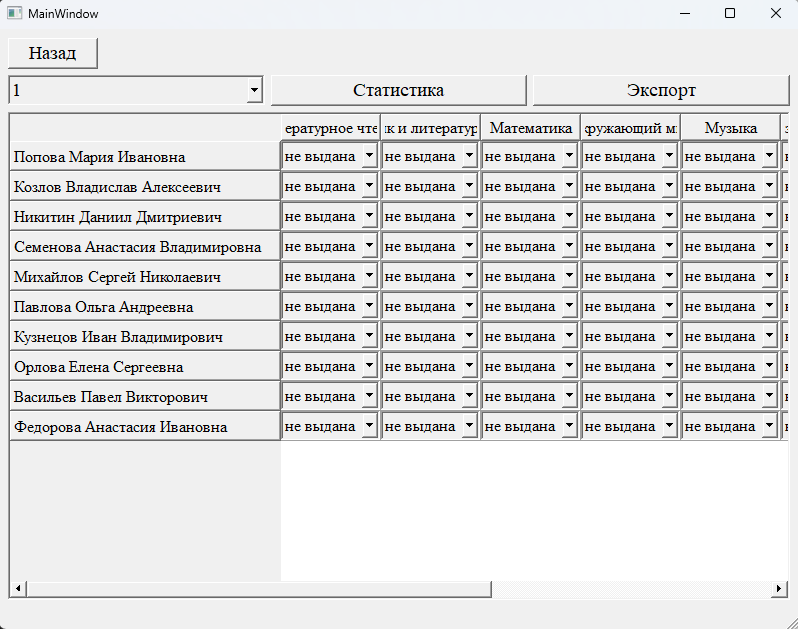


Рисунок 4 - Окно Ui\_library

Основные элементы интерфейса и события включают:

* Кнопка возврата “Назад”: возвращает в главное окно, вызывая класс модуля пользовательского интерфейса (Ui\_Main).
* Комбинированный список выбора группы: при изменении выбранного элемента вызывается функция load\_data\_to\_table, которая загружает соответствующие выбранному элементу данные в таблицу, вызывая методы модуля работы с базой данных для получения данных студентов и дисциплин.
* Кнопка “Cтатистика”: открывает окно статистики, вызывая класс модуля пользовательского интерфейса (Ui\_statistics).
* Кнопка экспорта данных: экспортирует данные библиотеки, вызывая модуль экспорта данных.

Класс **Ui\_statistics** предоставляет окно, представленное на рисунке 5, интерфейса для анализа данных. Этот класс предоставляет элементы для выбора типа статистики и отображения данных в таблице.

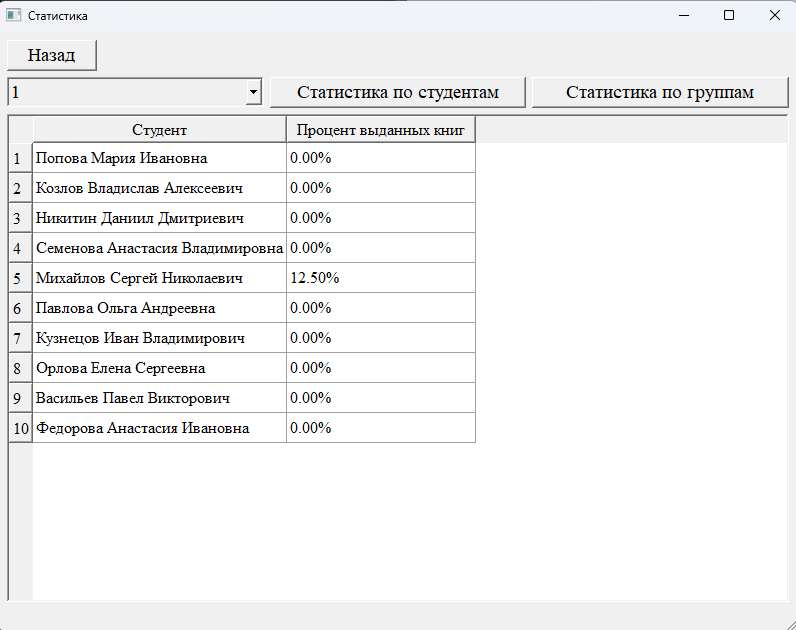


Рисунок 5 - Окно Ui\_statistics

Основные элементы интерфейса и события включают:

* Кнопка возврата “Назад”: возвращает в главное окно, вызывая класс модуля пользовательского интерфейса (Ui\_Main).
* Кнопка возврата “Статистика по студентам”: загружает соответствующий выбранному элементу в комбинированном список анализ данных статистики по студентам в таблицу, вызывая модуль анализа данных.
* Кнопка возврата “Статистика по группам”: загружает данные анализа данных статистики по группам в таблицу, вызывая модуль анализа данных.

Классы **Ui\_groups**, **Ui\_disciplines**, **Ui\_students**, **Ui\_books** имеют схожую структуру и функционал, направленные на управление соответствующими данными (группами, дисциплинами, студентами, книгами). Пример такого окна предоставлен на рисунке 6.

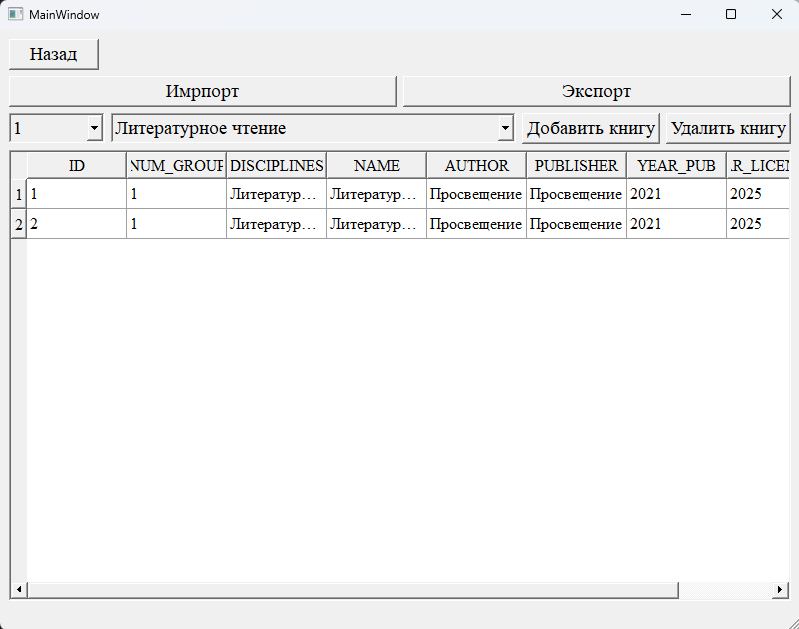


Рисунок 6- Окно Ui\_books

Основные элементы интерфейса и события включают:

* Комбинированный список выбора группы: при изменении выбранного элемента вызывается функция load\_data\_to\_table, которая загружает соответствующие выбранному элементу данные в таблицу, вызывая методы модуля работы с базой данных для получения данных о дисциплинах, студентах или книгах.
* Комбинированный список выбора дисциплины: при изменении выбранного элемента вызывается функция load\_data\_to\_table, которая загружает соответствующие выбранному элементу данные в таблицу, вызывая методы модуля работы с базой данных для получения данных студентов и дисциплин.
* Таблица данных: отображает соответствующие данные (группы, дисциплины, студенты, книги).
* Кнопка возврата “Назад”: возвращает в главное окно, вызывая класс модуля пользовательского интерфейса (Ui\_Main).
* Кнопка добавления (add\_button): добавляет новые записи в базу данных, вызывая методы модуля работы с базой данных.
* Кнопка удаления (delete\_button): удаляет выбранные записи из базы данных, вызывая методы модуля работы с базой данных.
* Кнопки импорта и экспорта данных (import\_button, export\_button): импортируют данные из XLSX файла и экспортируют данные в XLSX файл, вызывая методы модулей импорта и экспорта данных.

**Диалоговые окна**

Диалоговые окна предоставляют пользователю интерфейс для выполнения специфических задач.

Класс ConfirmDeleteDialog: используется для подтверждения удаления записи. Пользователь может подтвердить или отменить удаление. Пользователь может подтвердить или отменить удаление. В случае подтверждения вызов методов модуля работы с базой данных (DatabaseManager) для удаления записи.

Класс AddBookDialog: используется для добавления новой книги. Пользователь может ввести данные книги, которые затем передаются в модуль работы с базой данных для сохранения.

Таким образом, каждый класс интерфейса обеспечивает взаимодействие пользователя с различными модулями системы, такими как модуль работы с базой данных SQLite3, модуль импорта данных из XLSX и модуль экспорта данных в XLSX, обеспечивая полное и удобное управление данными библиотеки.

**Навигация между классами пользовательского интерфейса**

Для понимания структуры взаимодействия классов интерфейса и навигации между ними, была разработана диаграмма классов, представленная на рисунке 7. На диаграмме представлены основные классы пользовательского интерфейса и их взаимосвязи.

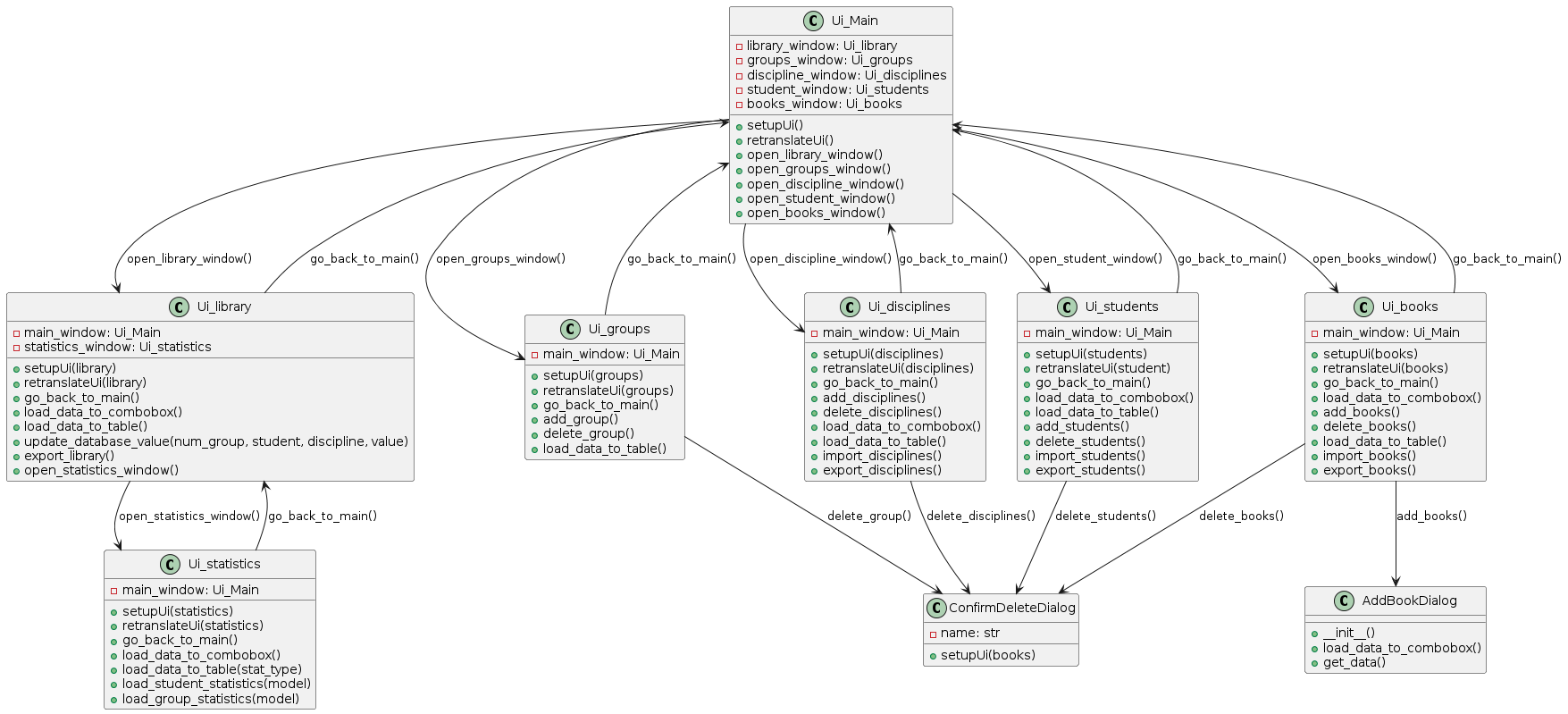


Рисунок 7 - Диаграмма классов

На диаграмме классов показаны основные классы интерфейса и их взаимосвязи. Класс Ui\_Main включает методы для открытия окон библиотечной деятельности (Ui\_library), групп (Ui\_groups), дисциплин (Ui\_disciplines), студентов (Ui\_students) и книг (Ui\_books).

Например, метод open\_library\_window() в классе Ui\_Main используется для перехода к окну управления библиотекой, представленному классом Ui\_library.

В каждом из этих классов предусмотрены методы для возврата к основному окну. Например, класс Ui\_library содержит метод go\_back\_to\_main(), который позволяет вернуться к главному окну интерфейса (Ui\_Main). Аналогичные методы имеются в классах Ui\_groups, Ui\_disciplines, Ui\_students и Ui\_books, что обеспечивает последовательную и логичную навигацию по приложению.

Важным аспектом навигации является также возможность перехода к специализированным диалоговым окнам для выполнения конкретных операций, таких как добавление или удаление записей. Например, класс Ui\_books содержит методы для открытия диалогов ConfirmDeleteDialog и AddBookDialog, что позволяет пользователю подтверждать удаление книг или добавлять новые книги в библиотеку.

Дополнительно, класс Ui\_library предоставляет возможность перехода к окну статистики (Ui\_statistics), где пользователь может просматривать и анализировать данные библиотеки. Переход осуществляется с помощью метода open\_statistics\_window(), а возврат к окну библиотеки возможен через метод go\_back() в классе Ui\_statistics.

Таким образом, навигация между классами пользовательского интерфейса организована через четко определенные методы, обеспечивающие плавный и интуитивный переход между различными частями приложения. Это позволяет пользователям легко ориентироваться в функционале программы и эффективно выполнять необходимые действия.

* 1. Модуль работы с базой данных SQLite3

Модуль работы с базой данных SQLite3 является ключевым компонентом разрабатываемого приложения для школьной библиотеки. Этот модуль отвечает за управление данными, обеспечивая надежное хранение, получение и манипулирование информацией о группах, студентах, дисциплинах и книгах. В данном разделе рассматривается структура и функциональность этого модуля.

Модуль работы с базой данных реализован в классе DatabaseManager, который отвечает за подключение к базе данных, создание необходимых таблиц и выполнение различных операций с данными. Ниже представлены основные функции класса DatabaseManager и их описание.

**Создание таблиц**

Функция createTable, представленная на рисунке 8, используется для создания физической модели данных в базе данных SQLite3.

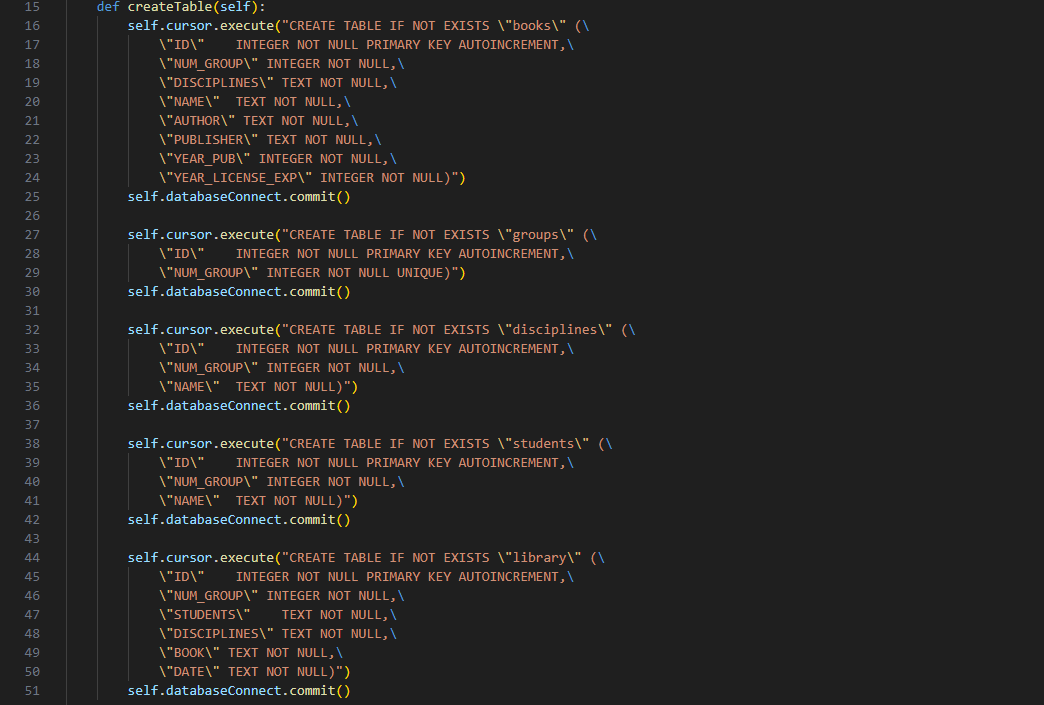


Рисунок 8 - Функция createTable

Функция создает таблицы books, groups, disciplines, students, library.

Каждое создание таблицы выполняется с помощью SQL-запроса **CREATE TABLE IF NOT EXISTS**, что гарантирует, что таблица будет создана только в случае отсутствия. После выполнения каждого запроса происходит фиксация изменений в базе данных с помощью **commit**, чтобы изменения вступили в силу.

**Добавление данных**

Добавление новых данных в таблицы базы данных осуществляется через функции класса DatabaseManager, такие как create\_group, create\_disciplines, create\_students и create\_books.

Пример функции для добавления группы представлен на рисунке 9.

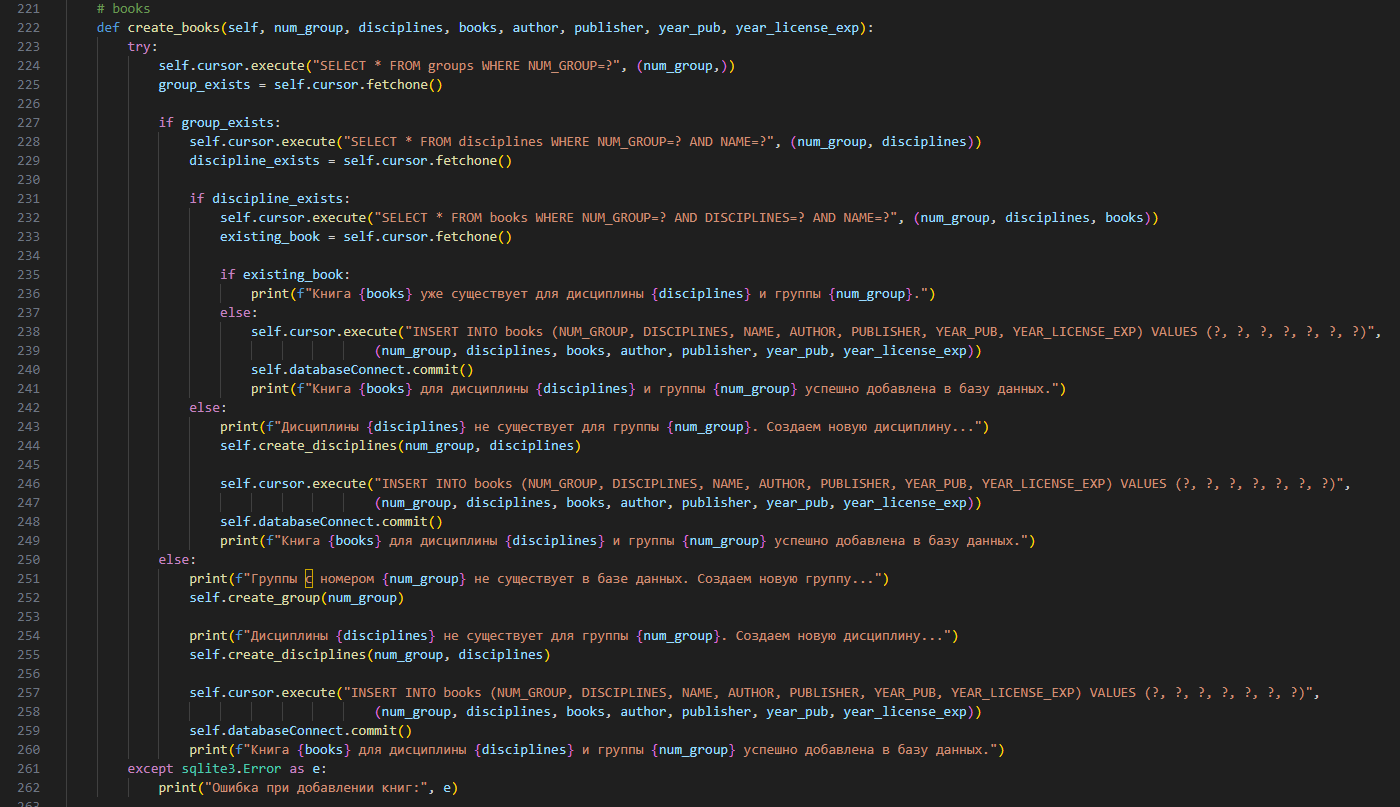


Рисунок 9 - Функция Сreate\_books

Одним из таких методов является create\_books. Этот метод принимает на вход информацию о книге, включая номер группы (num\_group), дисциплину (disciplines), название книги (books), автора (author), издателя (publisher), год публикации (year\_pub) и год истечения лицензии (year\_license\_exp).

Сначала метод проверяет существование группы и дисциплины в базе данных. Если группа и дисциплина уже существуют, метод проверяет, существует ли уже книга с такими параметрами. Если книга уже существует, выводится сообщение об этом. В противном случае новая группа добавляется в таблицу с помощью SQL-запроса **INSERT INTO** и фиксируются изменения в базе данных.

Если группа или дисциплина отсутствуют в базе данных, метод создает их сначала, а затем добавляет книгу. После каждой операции добавления данных метод фиксирует изменения в базе данных командой **commit**.

Подобным образом реализованы остальные функции по добавлению данных: create\_group, create\_disciplines и create\_students. Каждая из этих функций отвечает за добавление соответствующих сущностей в базу данных, при необходимости создавая их, если они отсутствуют.

**Удаление данных**

Удаление данных из таблиц базы данных осуществляется через функции delete\_group, delete\_disciplines, delete\_students и delete\_books. Пример функции для удаления группы представлен на рисунке 10.

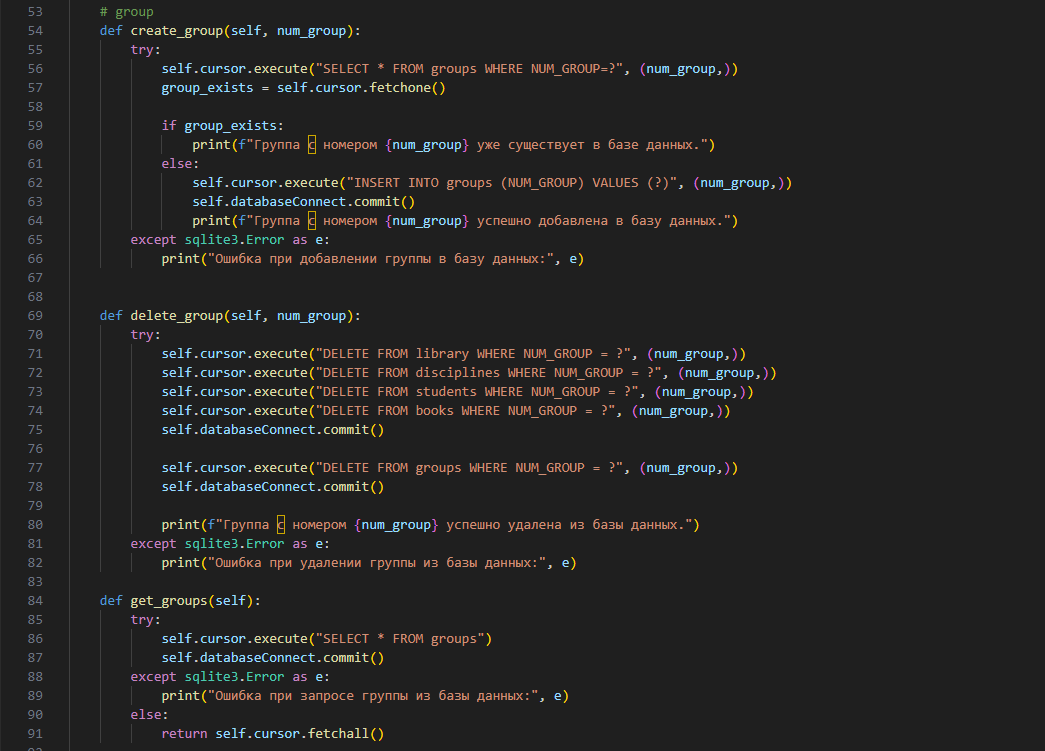


Рисунок 10 - Функция Delete\_group

Функция delete\_group используется для удаления группы из базы данных. Этот метод принимает номер группы (num\_group) в качестве аргумента. После чего удаляет все записи связанные с этой группой из всех таблиц базы данных с помощью SQL-запроса **DELETE**. После удаления данных метод фиксирует изменения в базе данных командой **commit**. После успешного удаления выводится сообщение об удалении группы.

Подобным образом реализованы остальные функции по удалению данных: delete\_disciplines, delete\_students, delete\_books. Каждая из этих функций отвечает за удаление соответствующих сущностей в базе данных, а также связанные с ним записи из других таблиц.

**Получение данных**

Пример функции для получения данных представлен на рисунке 11.

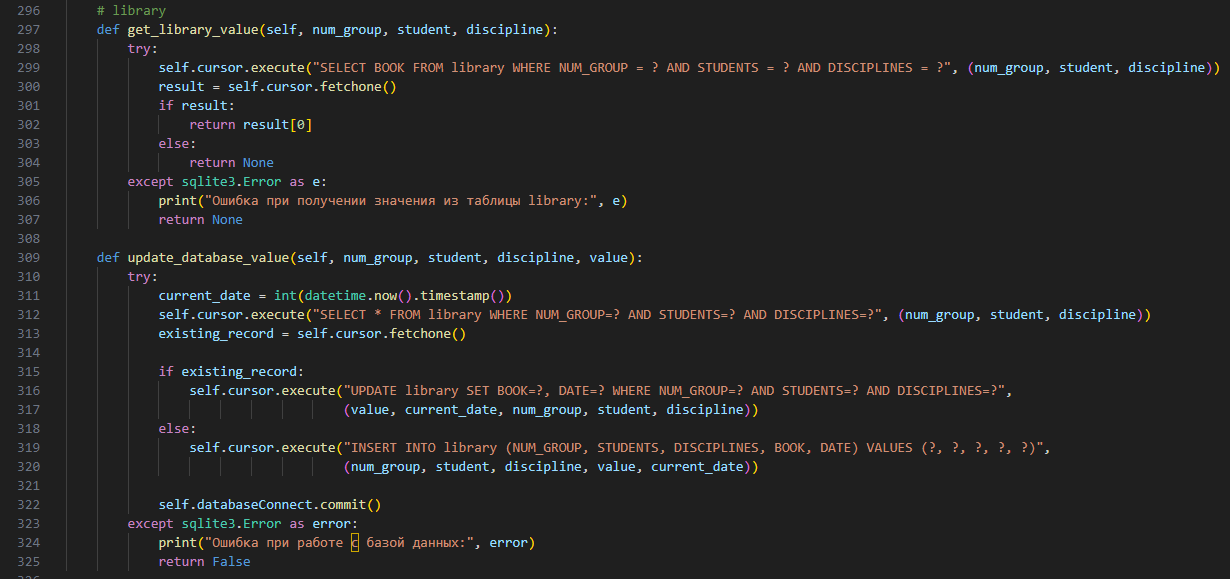


Рисунок 11 - Функция Get\_library\_value

Функция get\_library\_value предназначена для получения списка всех библиотечных записей из базы данных. Этот метод принимает на вход номер группы (num\_group), имя студента (student) и дисциплину (discipline), и возвращает значение с помощью запроса SQL-запрос **SELECT \* FROM** из соответствующей ячейки таблицы. Если значение найдено, оно возвращается; в противном случае возвращается None.

Примеры остальных функций по получению данных:

* get\_groups: возвращает список всех групп.
* get\_disciplines: возвращает список всех дисциплин.
* get\_disciplines\_by\_num\_group: возвращает список дисциплин для определенной группы.
* get\_students: возвращает список всех студентов.
* get\_students\_by\_num\_group: возвращает список студентов для определенной группы.
* get\_books: возвращает список всех книг.
* get\_books\_by\_num\_group\_and\_disciplines: возвращает список книг для определенной группы и дисциплины.

Каждая функция осуществляет запрос к базе данных и возвращает результат в виде списка кортежей или None в случае возникновения ошибки.

**Обновление данных**

Обновление данных в таблицах базы данных осуществляется через функции update\_database\_value. Эта функция, представленная на рисунке 12, обновляет записи в таблице library, добавляя или изменяя информацию о книгах, взятых студентами:

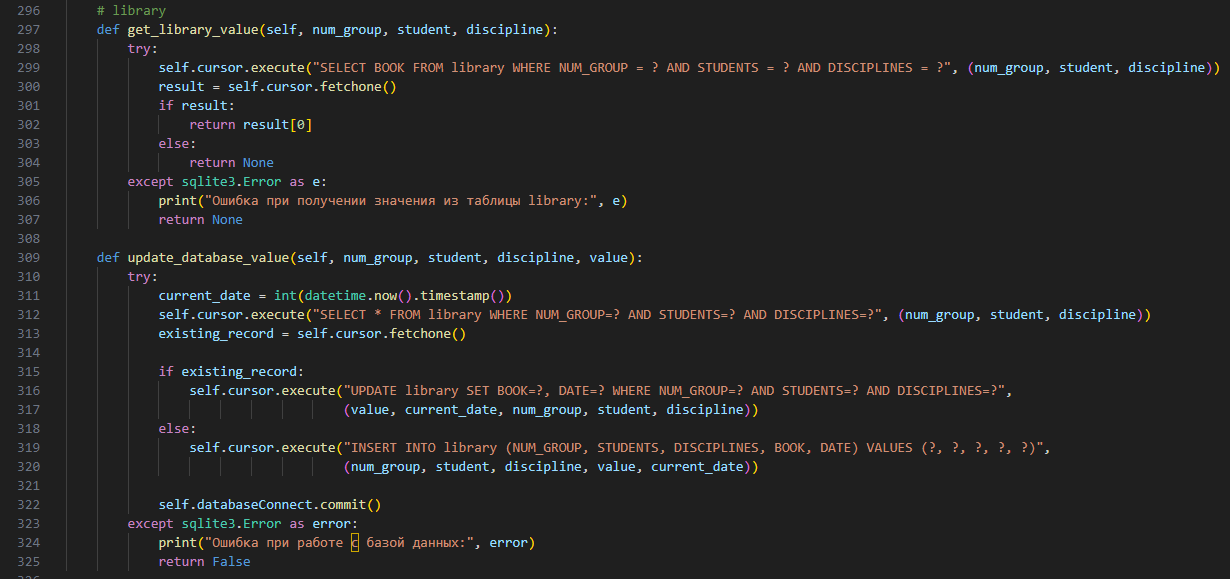


Рисунок 12 - Функция Update\_database\_val

Функция update\_database\_value предназначен для обновления значения таблицы library в базе данных. Функция принимает на вход параметры: номер группы (num\_group), имя студента (student), название дисциплины (discipline) и новое значение книги (value).

Сначала функция получает текущую дату и время в формате временной метки. Затем функция выполняет запрос к базе данных для проверки существующей записи о библиотечной записи для указанной группы, студента и дисциплины.

Если запись уже существует, метод обновляет значение библиотечной записи и дату в соответствии с переданными параметрами. Если запись не найдена, метод добавляет новую запись в таблицу library с указанными параметрами.

После завершения операции обновления или добавления данных метод фиксирует изменения в базе данных командой **commit**.

* 1. Модуль импорта данных из XLSX

В этом разделе описывается разработка модуля импорта данных из файлов формата XLSX в приложение для школьной библиотеки. Модуль позволяет загружать информацию о студентах, дисциплинах и книгах из электронных таблиц и сохранять её в базе данных SQLite3.

**Основная функциональность модуля**

Модуль импорта данных реализован с использованием библиотеки pandas для обработки данных из электронных таблиц и библиотеки PyQt5 для создания графического интерфейса пользователя.

Модуль состоит из класса ImportManager, который управляет процессом импорта. Класс содержит несколько методов для импорта различных типов данных: студентов, дисциплин и книг. Ниже приведена часть кода класса ImportManager.

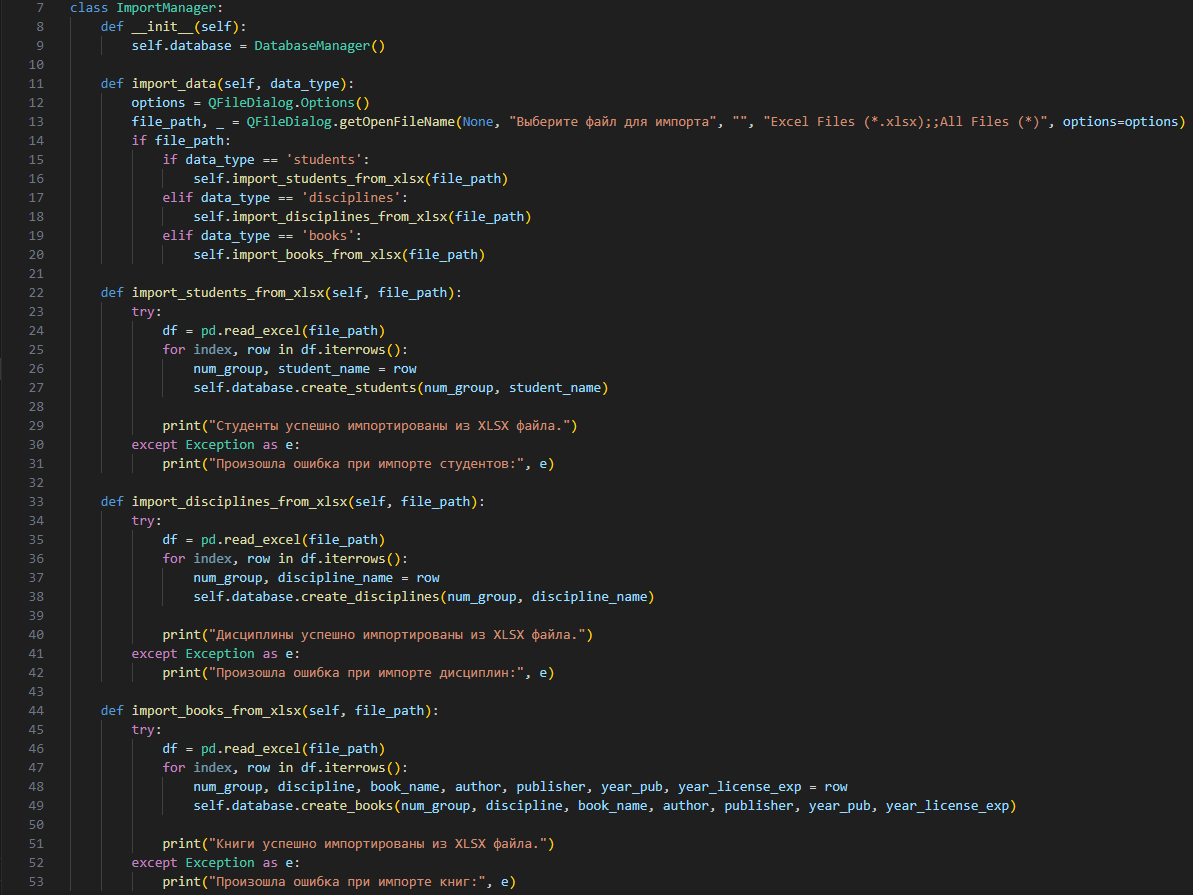


Рисунок 13 - Класс ImportManager

Класс ImportManager представляет собой менеджер импорта данных из файлов электронных таблиц в приложение.

Функция import\_data отвечает за общий процесс импорта данных. Функция принимает аргумент data\_type, определяющий тип данных, которые необходимо импортировать (студенты, дисциплины, книги). Затем Функция открывает диалоговое окно для выбора файла и, если пользователь выбрал файл, вызывает соответствующий метод для импорта данных этого типа.

Примером функции, реализующей импорт студентов из файла XLSX, является import\_students\_from\_xlsx. Функция считывает данные из файла, затем в цикле проходится по каждой строке данных, извлекает номер группы и имя студента, и передает их методу create\_students объекта database для сохранения в базе данных. По завершении импорта выводится сообщение об успешном выполнении операции или об ошибке, если возникла исключительная ситуация.

Оставшиеся функции (import\_disciplines\_from\_xlsx и import\_books\_from\_xlsx) выполняют аналогичные операции, но для импорта дисциплин и книг соответственно. Каждая из этих функций считывает данные из файла, извлекает необходимую информацию и сохраняют информацию в базе данных, а затем выводит сообщения о результате операции.

* 1. Модуль экспорта данных в XLSX

В этой главе описывается разработка модуля экспорта данных из базы данных SQLite3 в файлы формата XLSX для школьной библиотеки. Модуль позволяет выгружать информацию о студентах, дисциплинах, книгах и библиотечных данных в формат электронных таблиц, что обеспечивает удобство в управлении и анализе данных.

Модуль состоит из класса ExportManager, который управляет процессом экспорта. Класс содержит несколько методов для экспорта различных типов данных: студентов, дисциплин, книг и библиотечных данных. На рисунке 14 приведена часть кода класса ExportManager.

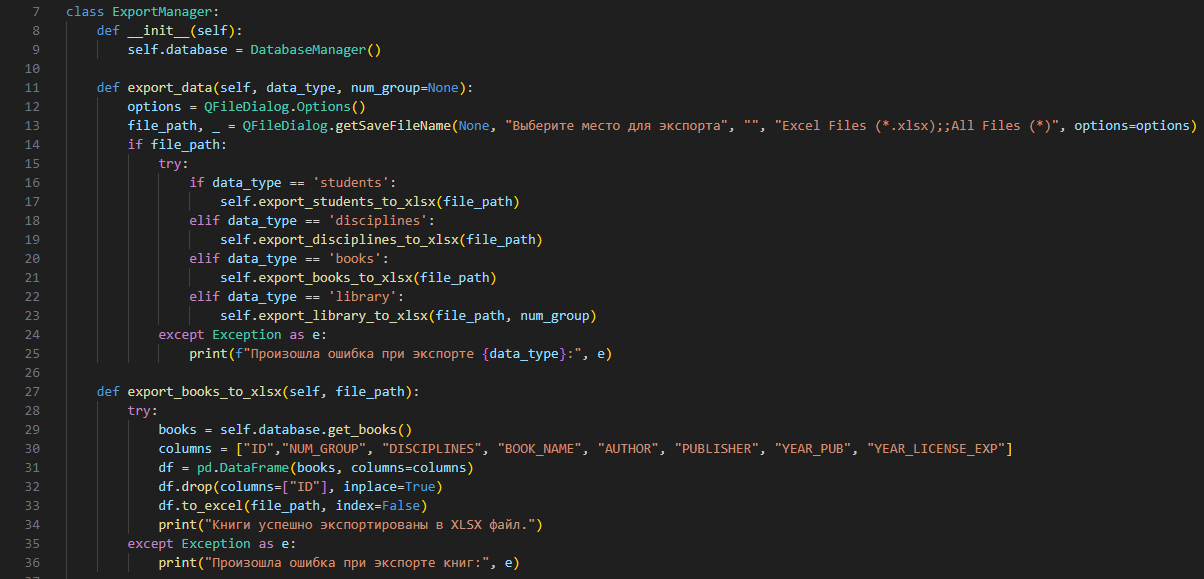


Рисунок 14 – Класс ExportManager

Функция export\_data отвечает за общий процесс экспорта данных. Она принимает два аргумента: data\_type, определяющий тип данных, которые необходимо экспортировать (студенты, дисциплины, книги, библиотека), и num\_group, который используется только для экспорта данных библиотеки. Открывается диалоговое окно для выбора места сохранения файла, и если путь к файлу был указан, вызывается соответствующий метод для экспорта данных выбранного типа. В случае возникновения ошибки во время экспорта, выводится сообщение об ошибке.

Примером функции, реализующей экспорт книг в файл XLSX, является export\_books\_to\_xlsx. Она получает данные книг из базы данных с помощью метода get\_books объекта database, создаёт DataFrame с этими данными, удаляет столбец "ID", и сохраняет DataFrame в файл XLSX. После успешного экспорта выводится сообщение об успешной операции, или об ошибке, если произошла исключительная ситуация.

Остальные функции (export\_disciplines\_to\_xlsx, export\_students\_to\_xlsx и export\_library\_to\_xlsx) выполняют аналогичные операции, но для экспорта дисциплин, студентов и данных библиотеки соответственно:

export\_disciplines\_to\_xlsx получает данные дисциплин из базы данных с помощью метода get\_disciplines, создаёт DataFrame, удаляет столбец "ID", и сохраняет данные в файл XLSX.

export\_students\_to\_xlsx получает данные студентов из базы данных с помощью метода get\_students, создаёт DataFrame, удаляет столбец "ID", и сохраняет данные в файл XLSX.

export\_library\_to\_xlsx получает данные дисциплин и студентов по номеру группы (num\_group), формирует структуру данных с именами студентов и значениями для каждой дисциплины, и сохраняет их в файл XLSX.

Каждая из этих функций обрабатывает свои данные специфическим образом, чтобы соответствовать требованиям экспорта для разных типов данных, и выводит сообщение о результате операции.

* 1. Модуль анализа данных

В этой главе рассматривается разработка модуля анализа данных, который является важным компонентом приложения для школьной библиотеки. Этот модуль предназначен для расчета статистики по выданным книгам, как для отдельных студентов, так и для целых групп. Основная задача модуля заключается в анализе данных, хранящихся в базе данных SQLite3, и предоставлении статистических результатов.

**Основная функциональность модуля**

Модуль анализа данных реализован в классе StatisticsManager.

Модуль состоит из класса StatisticsManager, который управляет процессом анализа данных. Класс содержит методы для расчета статистики для студентов и групп. На рисунке 15 приведена часть кода класса StatisticsManager.

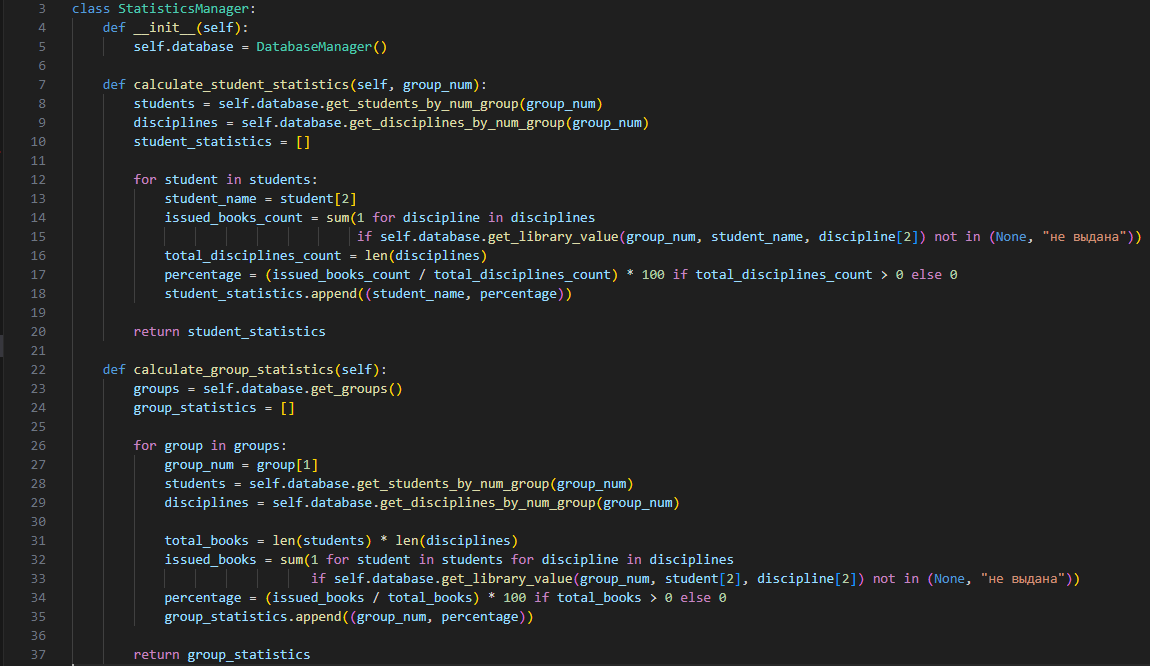


Рисунок 15 – Класс StatisticsManager

Функция calculate\_student\_statistics принимает номер группы и вычисляет статистику для каждого студента этой группы. Сначала функция получает список студентов и дисциплин данной группы из базы данных. Затем для каждого студента функция вычисляет количество выданных книг по его дисциплинам и общее количество дисциплин. По этим данным расчитывается процент выданных книг от общего количества дисциплин, и результат сохраняется в список student\_statistics.

Функция calculate\_group\_statistics вычисляет статистику по группам. Функция получает список всех групп из базы данных. Затем для каждой группы функция находит список студентов и дисциплин, вычисляет общее количество книг, которые должны быть выданы, и количество фактически выданных книг. По этим данным также расчитывается процент выданных книг от общего количества книг для каждой группы, и результат сохраняется в список group\_statistics.

Обе функции возвращают список кортежей, содержащих номер группы (или имя студента) и процент выданных книг для каждой группы (или студента) соответственно.

ЗАКЛЮЧЕНИЕ

В рамках данной выпускной квалификационной работы была разработана программа для школьной библиотеки, обеспечивающая работу с электронными таблицами формата XLSX и сохранение этих данных в базу данных SQLite3. Работа над проектом включала несколько ключевых этапов, каждый из которых способствовал достижению основной цели – создание эффективного инструмента для автоматизации библиотечной деятельности.

В первой главе был проведен всесторонний анализ предметной области. Был рассмотрен обзор существующих инструментов библиотечной деятельности. Это позволило сформулировать требования к разрабатываемому приложению. Кроме того, было описано, как данные из электронных таблиц могут быть считаны, подготовлены и проанализированы с использованием Python. Данный анализ предметной области позволил определить ключевые требования и инструменты к разработке программы.

Вторая глава была посвящена непосредственному проектированию и разработке приложения. В рамках проектирования модели базы данных была определена структура хранения данных, что обеспечило эффективную работу с ними. Описание структуры приложения позволило представить общую архитектуру решения и взаимодействие его компонентов. Были разработаны модули пользовательского интерфейса, работы с базой данных SQLite3, импорта и экспорта данных из формата XLSX, а также модуль анализа данных. Все эти компоненты были интегрированы в единое целое, обеспечивая функциональность, необходимую для работы школьной библиотеки.

В результате проделанной работы была создана программа, которая позволяет школьной библиотеке эффективно управлять данными о книгах, учениках и выдаче книг. Импорт данных из электронных таблиц позволяет быстро обновлять информацию, а возможность анализа и визуализации данных помогает библиотекарям принимать обоснованные решения. Экспорт данных обратно в XLSX обеспечивает удобство обмена информацией и создания отчетов.

Таким образом, разработанное приложение удовлетворяет всем требованиям, определенным на начальном этапе работы. Оно не только автоматизирует рутинные процессы, но и предоставляет новые возможности для анализа и управления данными, что в конечном итоге повышает эффективность работы школьной библиотеки.

СПИСОК ЛИТЕРАТУРЫ

1. Ковалева М. А., Волошин С. Б. Анализ данных. Учебное пособие. - М.: Мир науки, 2019
2. MyLib 1.0 // Лучший бесплатный софт URL: https://www.bestfree.ru/soft/office/accounting-books.php (дата обращения: 20.05.2024).
3. Учет книг // Allsoft URL: https://allsoft.ru/software/vendors/prostoy-soft/uchet-knig/ (дата обращения: 20.05.2024).
4. О программе «Школьная БИБЛИОТЕКА (учебники)» // Официальный сайт программы Школьная БИБЛИОТЕКА URL: https://school-libray.ru/ (дата обращения: 20.05.2024).
5. XLS vs. XLSX Files – What’s the Difference? // SPREADSHEETPLANET URL: https://spreadsheetplanet.com/xls-vs-xlsx-files/ (дата обращения: 20.05.2024).
6. Александер М., Куслейка Р. Excel 2019. Библия пользователя. - СПб.: ООО "Диалектика", 2019. - 1136 с.
7. Документация // openpyxl URL: https://openpyxl.readthedocs.io/en/stable/ (дата обращения: 21.05.2024).
8. Документация // xlrd URL: https://xlrd.readthedocs.io/en/latest/ (дата обращения: 21.05.2024).
9. Маккинни, У. Python и анализ данных. Первичная обработка данных с применением pandas, NumPy и Jupiter: справочник / У. Маккинни ; перевод с английского А. А. Слинкина. — 3-е изд. — Москва : ДМК Пресс, 2023.
10. Тарасов С. В. СУБД для программиста. Базы данных изнутри.. - M.,: СОЛОН-Пресс, 2015. - 320 с.
11. Джафаров, К. А. Экономические приложения теории вероятностей и математической статистики : учебное пособие / К. А. Джафаров, Л. В. Роева. — Новосибирск : НГТУ, 2022.
12. Букунов, С. В. Разработка приложений с графическим пользовательским интерфейсом на языке Python / С. В. Букунов, О. В. Букунова. — Санкт-Петербург : Лань, 2023.
13. PyQt vs. Tkinter — Which Should You Choose for Your Next GUI Project? // PythonGUIs URL: https://www.pythonguis.com/faq/pyqt-vs-tkinter/#the-pyqt-gui-framework (дата обращения: 22.05.2024).

ПРИЛОЖЕНИЕ

**ПРИЛОЖЕНИЕ А**

Рисунок 1 – QR-код, ведущий на страницу GitHub проекта



Таблица 1 – Сущности и атрибуты

|  |  |  |
| --- | --- | --- |
| Таблица books | | |
| Поле | Тип | Ограничения |
| ID | INTEGER | NOT NULL, PRIMARY KEY, AUTOINCREMENT |
| NUM\_GROUP | INTEGER | NOT NULL |
| DISCIPLINES | TEXT | NOT NULL |
| NAME | TEXT | NOT NULL |
| AUTHOR | TEXT | NOT NULL |
| PUBLISHER | TEXT | NOT NULL |
| YEAR\_PUB | INTEGER | NOT NULL |
| YEAR\_LICENSE\_EXP | INTEGER | NOT NULL |
| Таблица groups | | |
| ID | INTEGER | NOT NULL, PRIMARY KEY, AUTOINCREMENT |
| NUM\_GROUP | INTEGER | NOT NULL, UNIQUE |
| Таблица disciplines | | |
| ID | INTEGER | NOT NULL, PRIMARY KEY, AUTOINCREMENT |
| NUM\_GROUP | INTEGER | NOT NULL |
| NAME | TEXT | NOT NULL |
| Таблица students | | |
| ID | INTEGER | NOT NULL, PRIMARY KEY, AUTOINCREMENT |
| NUM\_GROUP | INTEGER | NOT NULL |
| NAME | TEXT | NOT NULL |
| Таблица library | | |
| ID | INTEGER | NOT NULL, PRIMARY KEY, AUTOINCREMENT |
| NUM\_GROUP | INTEGER | NOT NULL |
| STUDENTS | TEXT | NOT NULL |
| DISCIPLINES | TEXT | NOT NULL |
| BOOK | TEXT | NOT NULL |
| DATE | TEXT | NOT NULL |