

Оглавление

Реферат	2
Введение	3
Аналитическая часть	4
Постановка задачи	4
Общие сведения о БД СУБД	4
Типы БД	4
Иерархическая модель БД	4
Сетевая модель	5
Реляционная модель	5
Сравнение моделей	6
СУБД	7
Framework	8
Вывод	8
Конструкторская часть	9
Сценарий использования	9
Проектирование таблиц базы данных	9
Проектирование системы получения данных	22
Вывод	22
Технологическая часть	23
Выбор инструментов разработки	23
Реализация доступа к данным	23
Реализация обработки данных	24
Реализация доступа к данным	25
Интерфейс приложения	27
Требования к аппаратуре	27
Тестирование	28
Вывод	28
Заключение	29
Список литературы	30

Реферат

Расчетно-пояснительная записка содержит 30 стр., 5 рис., 4 листинга и 4 источника.

Ключевые слова: LaTeX, библиотека, проектирование, база данных, web, MS-SQL, веб-приложение, Web.

Курсовой проект представляет собой веб-приложение, в котором пользователь может создавать список литературы из книг, содержащихся в библиотеке.

В данной работе были рассмотрены различные подходы к хранению данных в БД.

Использовался язык программирования Python с библиотеками pyodbc и Dash.

Введение

При написании отчета в среде LaTeX, когда доходит дело до составления списка литературы, у многих возникают сложности связанные с тем, что в LaTeX не предусмотрен автоматический способ генерации элементов списка литературы, и все приходится писать вручную, что довольно мучительно и высока вероятность допустить ошибку. Поэтому в этой работе были поставлены задачи:

- 1) проанализировать существующие на рынке СУБД и выбрать наиболее подходящую для реализации библиотеки;
- 2) спроектировать базу данных для хранения информации об источниках информации;
- 3) реализовать web-приложение, позволяющее автоматически генерировать список литературы;
- 4) и наконец, провести тестирование и проверить работоспособность разработанного приложения.

Аналитическая часть

В данном разделе будут рассмотрены общие сведения о БД и СУБД, типы БД и используемые framework.

Формализация задачи

В соответствии с техническим заданием на курсовой проект, требуется создать базу данных для хранения книг в библиотеке, при этом должна быть предусмотрена возможность просмотра содержимого этой библиотеки.

Также требуется реализовать web-приложение, позволяющее создавать файл списка литературы и осуществлять просмотр содержимого библиотеки.

Общие сведения о БД СУБД

База данных представляет собой совокупность определенным образом организованных данных, которые хранятся в памяти вычислительной системы и отображают состояние объектов и их взаимосвязи в рассматриваемой предметной области.

Под системой управления базами данных понимается совокупность программных и языковых средств, предназначенных для создания и обработки БД.

Типы БД

Модель данных определяет логическую структуру БД и то, каким образом данные будут храниться, организовываться и обрабатываться.

Существует 3 типа моделей организации данных:

- иерархическая модель БД;
- сетевая модель БД;
- реляционная модель.

Иерархическая модель БД

Иерархическая модель БД представляет собой древовидную (иерархическую) структуру, состоящую из объектов (данных) различных уровней. Каждый объект может включать в себя несколько объектов более низкого уровня. Такие

объекты находятся в отношении предка (объект более близкий к корню) к потомку (объект более низкого уровня), при этом возможна ситуация, когда объект-предок имеет несколько потомков, тогда как у объекта-потомка обязателен только один предок.

Иерархической базой данных является файловая система, состоящая из корневого каталога, в котором имеется иерархия подкаталогов и файлов.

Связи записей реализуются в виде физических указателей с одной записи на другую. Основной недостаток иерархической структуры – невозможность реализовать отношения "многие-ко-многим" а также ситуации, в которых запись имеет несколько предков. [2]

Сетевая модель

Сетевая модель данных является расширением иерархического подхода. Разница между иерархической моделью данных и сетевой заключается в том, что в иерархических структурах запись-потомок должна иметь в точности одного предка, а в сетевой структуре у потомка может быть любое число предков. Записи в такой модели связаны списками с указателями.

Примером сетевой СУБД является IDMS (интегрированная система управления данными) от компании Computer Associates international Inc.

Популярность сетевой модели совпала с популярностью иерархической модели. Некоторые данные намного естественнее моделировать с несколькими предками для одного дочернего элемента. Сетевая модель позволяла моделировать отношения «многие-ко-многим».

И хотя эта модель широко применялась на практике, она так и не стала доминантной по двум основным причинам. Во-первых, компания IBM решила не отказываться от иерархической модели в расширениях для своих продуктов, таких как IMS и DL/I. Во-вторых, через некоторое время ее сменила реляционная модель, предлагавшая более высокоуровневый, декларативный интерфейс. [2]

Реляционная модель

В реляционной модели, в отличие от иерархической или сетевой, не существует физических отношений. Вся информация хранится в виде таблиц (отношений), состоящих из рядов и столбцов. А данные двух таблиц связаны общими столб-

цами, а не физическими ссылками или указателями. Объекты и их отношения представлены таблицами.

В реляционных моделях нет необходимости просматривать все указатели, что облегчает выполнение запросов на выборку информации по сравнению с сетевыми и иерархическими БД. Это одна из основных причин, почему реляционная модель оказалась более удобна.

Самые распространенные реляционные СУБД: MySQL, PostgreSQL, Access, Oracle, DB2, MS-SQL Server, SQLite. Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

- 1) каждый элемент таблицы — один элемент данных;
- 2) все элементы в одном столбце имеют одинаковый тип;
- 3) каждый столбец имеет уникальное имя;
- 4) одинаковые строки (записи, кортежи) в таблице отсутствуют;
- 5) порядок следования строк и столбцов может быть произвольным.

Каждое поле содержит одну характеристику объекта предметной области. В записи собраны сведения об одном экземпляре этого объекта.

Некоторые поля могут быть определены как ключевые. Это значит, что для ускорения поиска конкретных значений будет использоваться индексация. Когда поля двух различных таблиц получают данные из одного набора, можно использовать оператор JOIN для выбора связанных записей двух таблиц, сопоставив значения полей. Такие действия можно расширить до объединения нескольких полей в нескольких таблицах. Поскольку отношения здесь определяются только временем поиска, реляционные базы данных классифицируются как динамические системы.

Сравнение моделей

Иерархическая модель данных поддерживает отношения типа «один-к-одному» или «один-ко-многим». Она позволяет быстро получать данные, но не отличается гибкостью. Иногда роль элемента (родителя или потомка) неясна и не подходит для иерархической модели.

Вторая, сетевая модель данных, имеет более гибкую структуру, чем иерархическая, и поддерживает отношение «многие ко многим». Но быстро становится слишком сложной и неудобной для управления.

Третья модель – реляционная – более гибкая, чем иерархическая и проще для управления, чем сетевая. Реляционная модель сегодня используется чаще всего, так как имеет множество преимуществ, таких как:

- 1) простота использования;
- 2) гибкость;
- 3) независимость данных;
- 4) безопасность;
- 5) простота практического применения;
- 6) слияние данных;
- 7) целостность данных.

В связи с этим, далее будет рассматриваться реляционная модель. Осталось только рассмотреть систему управления такой моделью.

СУБД

Среди самых популярных систем управления реляционными базами данных:

- 1) MS-SQL Server;
- 2) MySQL;
- 3) PostgreSQL;
- 4) SQLite.

В данном проекте будет рассмотрена СУБД MS-SQL Server.

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

SQL Server был создан компанией Microsoft. Первая версия вышла в 1987 году. А текущей версией является версия 16, которая вышла в 2016 году и которая будет использоваться в текущем руководстве.

SQL Server долгое время был исключительно системой управления базами данных для Windows, однако начиная с версии 16 эта система доступна и на Linux [3].

Преимущества:

- 1) **Производительность.** SQL Server работает очень быстро.
- 2) **Надежность и безопасность.** SQL Server предоставляет шифрование данных.
- 3) **Простота.** С данной СУБД относительно легко работать и вести администрирование.

Framework

SQL Server совместима со множеством фреймворков, которые содержат в себе требуемые методы обращения к БД. Среди возможных вариантов для использования в проекте была выбрана библиотека pyodbc [3].

pyodbc — это программная библиотека на языке Python для работы с реляционными СУБД с применением технологии ORM. Служит для синхронизации объектов Python и записей реляционной базы данных. [?]

В качестве web-framework был выбран Dash, который предоставляет все необходимые инструменты для создания подобного проекта, так как предоставляет возможность для написания как frontend, так и backend для полноценного запуска приложения. [?]

Вывод

В результате проведенного анализа в качестве модели данных была выбрана реляционная модель, в качестве СУБД – MS-SQL Server.

Таким образом, подобрав необходимый набор инструментов для реализации web-приложения, можно приступить к проектированию решения поставленной задачи.

Конструкторская часть

В соответствии с техническим заданием и аналитическим разделом, должно быть получено полноценное приложение для взаимодействия с БД.

Сценарий использования

На рисунке 1 представлен сценарий использования приложения пользователем.

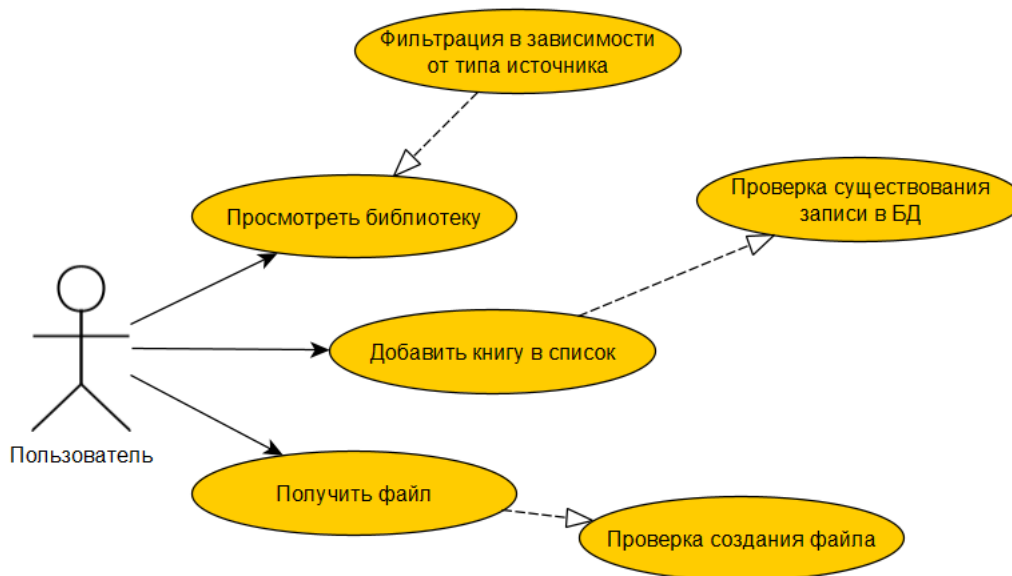


Рис. 1: Сценарий использования приложения

Проектирование таблиц базы данных

База данных приложения состоит из следующих таблиц:

- 1) таблица издателей **publisher**;
- 2) таблица авторов **author**;
- 3) таблица книг **book**;
- 4) таблица нормативно-правовых документов **regulatory_doc**;
- 5) таблица нормативно-технических документов **regulatory_tech_doc**;
- 6) таблица авторских свидетельств, патентов **patent**;
- 7) таблица информационных листов **info_sheet**;

- 8) таблица многотомных изданий **multi _volume**;
- 9) таблица диссертаций **not _published _yet**;
- 10) таблица электронных ресурсов локального доступа (CD) **elec _local**;
- 11) таблица электронных ресурсов удаленного доступа (Internet) **elec _internet**;
- 12) таблица статей из книг **book _article**;
- 13) таблица статей из газет **newspaper _article**;
- 14) таблица рецензий **review**;

Таблица publisher

Таблица, содержащая информацию об издателях. С помощью связи «один-к-одному» связана почти со всеми остальными таблицами БД.

Таблица содержит следующие поля:

- 1) id - целочисленное поле, идентификационный номер издателя;
- 2) name - символьное поле, название издателя;
- 3) place _of _ed – город издательства;
- 4) extra _ed _info – дополнительная информация об издательстве.

Таблица author

Таблица, содержащая информацию об авторах. С помощью связи «один-к-одному» связана почти со всеми остальными таблицами БД.

Таблица содержит следующие поля:

- 1) id - целочисленное поле, идентификационный номер автора;
- 2) name - символьное поле, имя автора;
- 3) extra _info – символьное поле, дополнительная информация об авторе.

Таблица book

Позволяет однозначно описать источник информации, являющийся книгой. Имеет связь «один-ко-многим» с таблицами author и publisher.

Таблица содержит следующие поля:

- 1) id - целочисленное поле, идентификационный номер источника;
- 2) name - символьное поле, название источника;
- 3) author_amount - целочисленное поле, обозначающее количество авторов у источника;
- 4) first_author - целочисленное поле, идентификационный номер первого автора;
- 5) second_author - целочисленное поле, идентификационный номер второго автора;
- 6) third_author - целочисленное поле, идентификационный номер третьего автора;
- 7) fourth_author - целочисленное поле, идентификационный номер четвертого автора;
- 8) extra_info - символьное поле, содержащее дополнительную информацию о источнике;
- 9) publisher_id - целочисленное поле, идентификационный номер издателя;
- 10) year - целочисленное поле, обозначающее год издания источника;
- 11) pages - целочисленное поле, обозначающее количество страниц источника.

Таблица regulatory_doc

Позволяет однозначно описать источник информации, являющийся нормативно-правовым документом. Имеет связь «один-ко-многим» с таблицей publisher.

Таблица содержит следующие поля:

- 1) id - целочисленное поле, идентификационный номер источника;
- 2) name - символьное поле, название источника;
- 3) extra_info - символьное поле, содержащее дополнительную информацию о источнике;
- 4) publisher_id - целочисленное поле, идентификационный номер издателя;
- 5) year - целочисленное поле, обозначающее год издания источника;
- 6) journal_id - целочисленное поле, обозначающее номер журнала, в котором был опубликован документ;
- 7) first_page - целочисленное поле, обозначающее первую страницу документа;
- 8) last_page - целочисленное поле, обозначающее последнюю страницу документа.

Таблица regulatory_tech_doc

Позволяет однозначно описать источник информации, являющийся нормативно-техническим документом. Имеет связь «один-ко-многим» с таблицей publisher.

Таблица содержит следующие поля:

- 1) id - целочисленное поле, идентификационный номер источника;
- 2) name - символьное поле, название источника;
- 3) extra_info - символьное поле, содержащее дополнительную информацию о источнике;
- 4) date - поле даты, указывающее дату принятия документа;
- 5) publisher_id - целочисленное поле, идентификационный номер издателя;
- 6) year - целочисленное поле, обозначающее год издания источника;
- 7) pages - целочисленное поле, обозначающее количество страниц документа.

Таблица patent

Позволяет однозначно описать источник информации, являющийся патентом. Имеет связь «один-ко-многим» с таблицами author и publisher.

Таблица содержит следующие поля:

- 1) id - целочисленное поле, идентификационный номер источника;
- 2) name - символьное поле, название источника;
- 3) author_amount - целочисленное поле, обозначающее количество авторов у источника;
- 4) first_author - целочисленное поле, идентификационный номер первого автора;
- 5) second_author - целочисленное поле, идентификационный номер второго автора;
- 6) third_author - целочисленное поле, идентификационный номер третьего автора;
- 7) fourth_author - целочисленное поле, идентификационный номер четвертого автора;
- 8) publisher_id - целочисленное поле, идентификационный номер издателя;
- 9) journal_id - целочисленное поле, обозначающее номер журнала, в котором был опубликован документ;
- 10) year - целочисленное поле, обозначающее год издания источника;
- 11) pages - целочисленное поле, обозначающее количество страниц источника.

Таблица info_sheet

Позволяет однозначно описать источник информации, являющийся информационным листком. Имеет связь «один-ко-многим» с таблицами author и publisher.

Таблица содержит следующие поля:

- 1) id - целочисленное поле, идентификационный номер источника;

- 2) name - символьное поле, название источника;
- 3) author_amount - целочисленное поле, обозначающее количество авторов у источника;
- 4) first_author - целочисленное поле, идентификационный номер первого автора;
- 5) second_author - целочисленное поле, идентификационный номер второго автора;
- 6) third_author - целочисленное поле, идентификационный номер третьего автора;
- 7) fourth_author - целочисленное поле, идентификационный номер четвертого автора;
- 8) publisher_id - целочисленное поле, идентификационный номер издателя;
- 9) year - целочисленное поле, обозначающее год издания источника;
- 10) pages - целочисленное поле, обозначающее количество страниц источника.

Таблица multi_volume

Позволяет однозначно описать источник информации, являющийся многотомным изданием. Имеет связь «один-ко-многим» с таблицами author и publisher.

Таблица содержит следующие поля:

- 1) id - целочисленное поле, идентификационный номер источника;
- 2) name - символьное поле, название источника;
- 3) author_amount - целочисленное поле, обозначающее количество авторов у источника;
- 4) first_author - целочисленное поле, идентификационный номер первого автора;
- 5) second_author - целочисленное поле, идентификационный номер второго автора;

- 6) `third_author` - целочисленное поле, идентификационный номер третьего автора;
- 7) `fourth_author` - целочисленное поле, идентификационный номер четвертого автора;
- 8) `extra_info` - символьное поле, содержащее дополнительную информацию о источнике;
- 9) `publisher_id` - целочисленное поле, идентификационный номер издателя;
- 10) `year` - целочисленное поле, обозначающее год издания источника;
- 11) `amount_of_vols` - целочисленное поле, обозначающее количество томов источника.

Таблица `not_published_yet`

Позволяет однозначно описать источник информации, являющийся диссертацией. Имеет связь «один-ко-многим» с таблицами `author` и `publisher`.

Таблица содержит следующие поля:

- 1) `id` - целочисленное поле, идентификационный номер источника;
- 2) `name` - символьное поле, название источника;
- 3) `author` - целочисленное поле, идентификационный номер автора;
- 4) `date` - поле даты, указывающее дату защиты документа;
- 5) `date` - поле даты, указывающее дату утверждения документа;
- 6) `extra_info` - символьное поле, содержащее дополнительную информацию о источнике;
- 7) `publisher_id` - целочисленное поле, идентификационный номер издателя;
- 8) `year` - целочисленное поле, обозначающее год издания источника.

Таблица `elec_local`

Позволяет однозначно описать источник информации, являющийся электронным ресурсом локального доступа (CD). Имеет связь «один-ко-многим» с таблицами author и publisher.

Таблица содержит следующие поля:

- 1) id - целочисленное поле, идентификационный номер источника;
- 2) name - символьное поле, название источника;
- 3) author_amount - целочисленное поле, обозначающее количество авторов у источника;
- 4) first_author - целочисленное поле, идентификационный номер первого автора;
- 5) second_author - целочисленное поле, идентификационный номер второго автора;
- 6) third_author - целочисленное поле, идентификационный номер третьего автора;
- 7) fourth_author - целочисленное поле, идентификационный номер четвертого автора;
- 8) extra_info - символьное поле, содержащее дополнительную информацию о источнике;
- 9) type - символьное поле, обозначающее тип ресурса;
- 10) publisher_id - целочисленное поле, идентификационный номер издателя;
- 11) year - целочисленное поле, обозначающее год издания источника;
- 12) material_desc - символьное поле, обозначающее информацию о материале источника.

Таблица elec_internet

Позволяет однозначно описать источник информации, являющийся электронным ресурсом удаленного доступа (Internet). Имеет связь «один-ко-многим» с таблицами author и publisher.

Таблица содержит следующие поля:

- 1) id - целочисленное поле, идентификационный номер источника;
- 2) name - символьное поле, название источника;
- 3) author_amount - целочисленное поле, обозначающее количество авторов у источника;
- 4) first_author - целочисленное поле, идентификационный номер первого автора;
- 5) second_author - целочисленное поле, идентификационный номер второго автора;
- 6) third_author - целочисленное поле, идентификационный номер третьего автора;
- 7) fourth_author - целочисленное поле, идентификационный номер четвертого автора;
- 8) extra_info - символьное поле, содержащее дополнительную информацию о источнике;
- 9) publisher_id - целочисленное поле, идентификационный номер издателя;
- 10) year - целочисленное поле, обозначающее год издания источника;
- 11) url - символьное поле, содержащее ссылку на ресурс.

Таблица book_article

Позволяет однозначно описать источник информации, статьей из книги. Имеет связь «один-ко-многим» с таблицами author, book и publisher.

Таблица содержит следующие поля:

- 1) id - целочисленное поле, идентификационный номер источника;
- 2) name - символьное поле, название источника;
- 3) author_amount - целочисленное поле, обозначающее количество авторов у источника;

- 4) first_author - целочисленное поле, идентификационный номер первого автора;
- 5) second_author - целочисленное поле, идентификационный номер второго автора;
- 6) third_author - целочисленное поле, идентификационный номер третьего автора;
- 7) fourth_author - целочисленное поле, идентификационный номер четвертого автора;
- 8) extra_info - символьное поле, содержащее дополнительную информацию о источнике;
- 9) book_id - целочисленное поле, идентификационный номер книги;
- 10) year - целочисленное поле, обозначающее год издания источника;
- 11) page - целочисленное поле, обозначающее номер страницы статьи.

Таблица newspaper_article

Позволяет однозначно описать источник информации, являющийся статьей из газеты. Имеет связь «один-ко-многим» с таблицами author, book и publisher.

Таблица содержит следующие поля:

- 1) id - целочисленное поле, идентификационный номер источника;
- 2) name - символьное поле, название источника;
- 3) author_amount - целочисленное поле, обозначающее количество авторов у источника;
- 4) first_author - целочисленное поле, идентификационный номер первого автора;
- 5) second_author - целочисленное поле, идентификационный номер второго автора;
- 6) third_author - целочисленное поле, идентификационный номер третьего автора;

- 7) `fourth_author` - целочисленное поле, идентификационный номер четвертого автора;
- 8) `extra_info` - символьное поле, содержащее дополнительную информацию о источнике;
- 9) `newspaper_name` - символьное поле, содержащее название газеты;
- 10) `newspaper_year` - целочисленное поле, обозначающее год издания газеты;
- 11) `newspaper_number` - целочисленное поле, обозначающее номер газеты;
- 12) `year` - целочисленное поле, обозначающее год издания источника;
- 13) `page` - целочисленное поле, обозначающее номер страницы статьи.

Таблица review

Позволяет однозначно описать источник информации, являющийся рецензией. Имеет связь «один-ко-многим» с таблицами `author`, `book` и `publisher`.

Таблица содержит следующие поля:

- 1) `id` - целочисленное поле, идентификационный номер источника;
- 2) `name` - символьное поле, название источника;
- 3) `author_amount` - целочисленное поле, обозначающее количество авторов у источника;
- 4) `first_author` - целочисленное поле, идентификационный номер первого автора;
- 5) `second_author` - целочисленное поле, идентификационный номер второго автора;
- 6) `third_author` - целочисленное поле, идентификационный номер третьего автора;
- 7) `fourth_author` - целочисленное поле, идентификационный номер четвертого автора;

- 8) extra_info - символьное поле, содержащее дополнительную информацию о источнике;
- 9) journal_name - символьное поле, содержащее название журнала;
- 10) journal_year - целочисленное поле, обозначающее год издания журнала;
- 11) journal_number - целочисленное поле, обозначающее номер журнала;
- 12) year - целочисленное поле, обозначающее год издания источника;
- 13) page - целочисленное поле, обозначающее номер страницы статьи.

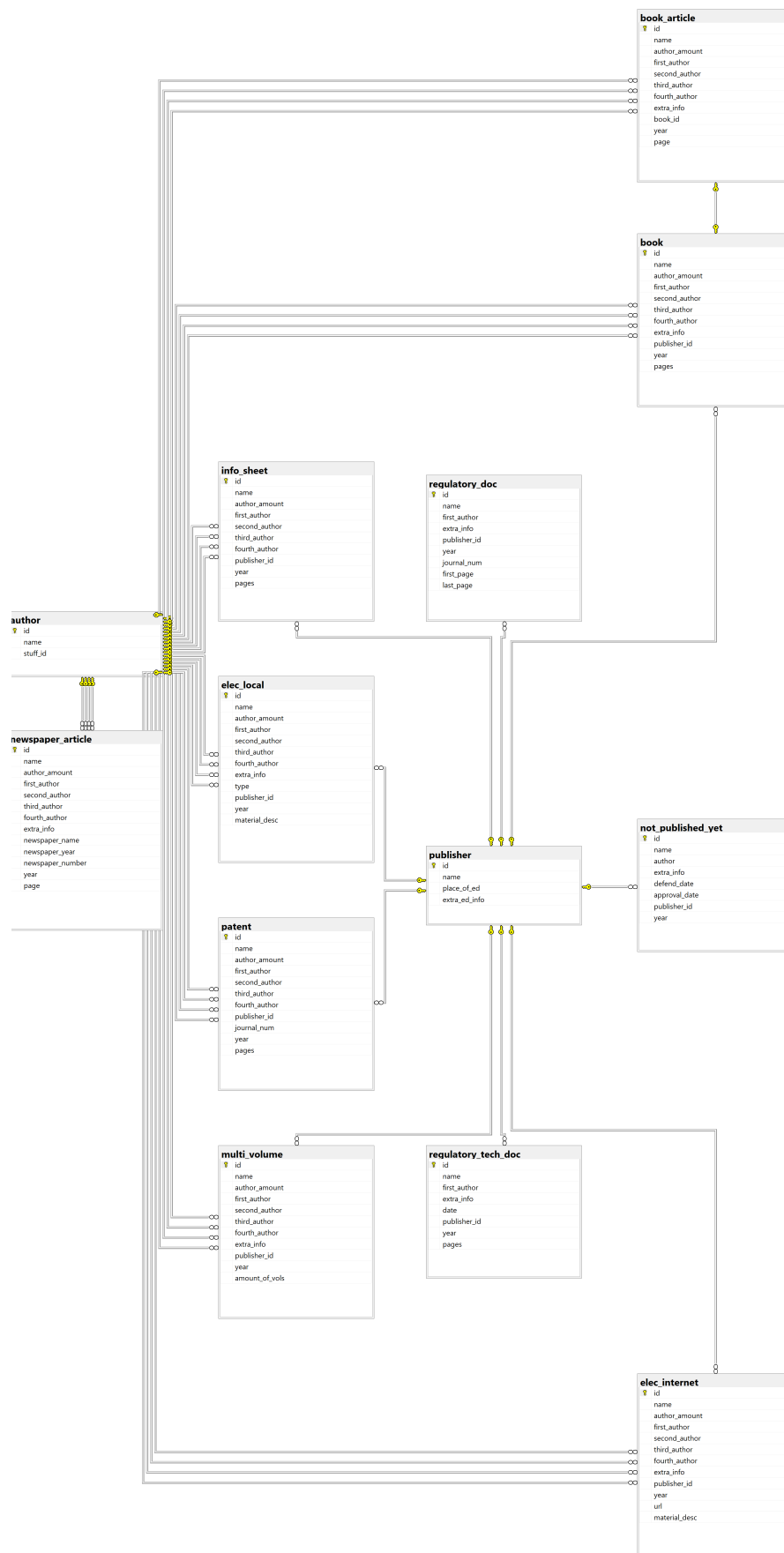


Рис. 2: Диаграмма БД

Проектирование системы доступа к данным

Для обеспечения доступа к данным будем использовать библиотеку `pyodbc` для Python, отправляя SQL запросы в нашу БД.

Создание web-интерфейса будет реализовано с помощью фреймворка Dash. Приложения Dash должны состоять из двух частей: `layout` (то, что будет выведено на экран) и самого кода программы.

Фреймворк Dash предоставляет собой набор данных базовых инструментов для реализации web-приложения, поэтому интерфейс будет создан с помощью уже готовых средств.

Вывод

Была разработана модель приложения, дающего возможность генерировать список литературы для среды LaTeX.

Технологическая часть

После проектирования структуры поставленной задачи, требуется реализовать набор функций, необходимый для создания web-приложения, а также конкретизировать полный список инструментов, используемых для запуска приложения.

Выбор инструментов разработки

В ходе реализации были использованы следующие технологии и средства:

- язык программирования Python;
- СУБД MS-SQL;
- библиотека pyodbc;
- фреймворк Dash.

Такой набор инструментов был выбран в связи с тем, что для каждого из элементов предусмотрено взаимодействие с другими. Помимо этого, данные инструменты полностью выполняют задачи, необходимые для реализации проекта.

Для разработки и тестирования приложения был использован компьютер с операционной системой Windows 10.

Реализация доступа к данным

Для работы с базой данных для начала требуется установить соединение.

Листинг 1: Подключение к БД

```
print("Идет загрузка БД, пожалуйста, подождите...")

cnxn = pyodbc.connect(driver='{SQL Server}', Server='DESKTOP-IEVL6PF', database='
    course_project',
    trusted_connection='yes')
cursor = cnxn.cursor()

print("БД была успешно загружена")

cursor.execute("select * from book")
books = cursor.fetchall()
```

Ниже представлено получение данных для просмотра в web-приложении в зависимости от выбранной категории.

```

@app.callback(Output('table', 'data'),
[Input('input3', 'value')])
)
def load_table(value):
    global cursor, columns_of_books

    if value == '1':

        cursor.execute("select id, name, first_author, year, extra_info from book")
        books = cursor.fetchall()

        return [dict(zip(columns_of_books, d)) for d in books]

    if value == '2':

        cursor.execute("select id, name, first_author, year, extra_info from regulatory_doc")
        books = cursor.fetchall()

        return [dict(zip(columns_of_books, d)) for d in books]
    ...

```

Реализация обработки данных

Для работы с данными потребовалось реализовать несколько различных функций из-за отличий в правилах оформления для каждого источника.

Ниже представлена функция обработки книг в правильный формат.

Листинг 3: Обработка данных

```

def print_books(input_str):
    global cursor, str_to_write, counter

    cursor.execute("select * from book where name = '{}'".format(input_str))
    book = cursor.fetchall()
    str_to_write += "\n\\bibitem{{{litlink}}}" ".format(counter)

    counter += 1
    amount_of_authors = book[0][2]

    if amount_of_authors == 1:
        str_to_write += u"{{}. {{} [Текст]".format(book[0][3], book[0][1])
    if book[0][7] != None:
        str_to_write += u": {{} / {{} - {{}: {{}, {{}. - {{} с.".format(book[0][7], book[0][3],
            book[0][8], book[0][10],
            book[0][11], book[0][12])
    else:
        str_to_write += u" / {{} - {{}: {{}, {{}. - {{} с.".format(book[0][3], book[0][8], book
            [0][10],
            book[0][11], book[0][12])

```



```

elif amount_of_authors == 2:
    str_to_write += u"{}. {} [Текст]".format(book[0][3], book[0][1])
    if book[0][7] != None:
        str_to_write += u": {} / {}, {} - {}: {}, {}. - {} c.".format(book[0][7], book
            [0][3], book[0][4], book[0][8], book[0][10],
            book[0][11], book[0][12])
    else:
        str_to_write += u" / {}, {} - {}: {}, {}. - {} c.".format(book[0][3], book[0][4],
            book[0][8], book[0][10],
            book[0][11], book[0][12])

elif amount_of_authors == 3:
    str_to_write += u"{}. {} [Текст]".format(book[0][3], book[0][1])
    if book[0][7] != None:
        str_to_write += u": {} / {}, {}, {} - {}: {}, {}. - {} c.".format(book[0][7],
            book[0][3], book[0][4],
            book[0][5], book[0][8], book[0][10],
            book[0][11], book[0][12])
    else:
        str_to_write += u" / {}, {}, {} - {}: {}, {}. - {} c.".format(book[0][3], book
            [0][4], book[0][5],
            book[0][8], book[0][10],
            book[0][11], book[0][12])
elif amount_of_authors == 4:
    str_to_write += u"{}. {} [Текст]".format(book[0][3], book[0][1])
    if book[0][7] != None:
        str_to_write += u": {} / {}, {}, {}, {} - {}: {}, {}. - {} c.".format(book[0][7],
            book[0][3], book[0][4],
            book[0][5], book[0][6], book[0][8], book[0][10],
            book[0][11], book[0][12])
    else:
        str_to_write += u" / {}, {}, {}, {} - {}: {}, {}. - {} c.".format(book[0][3], book
            [0][4], book[0][5],
            book[0][6], book[0][8], book[0][10],
            book[0][11], book[0][12])

```

Frontend-разработка

Чтобы обеспечить доступ к данным нужно создать форму, позволяющую отображать записи в таблицах. В Dash для этого используется класс `dash_table.DataTable`. Поскольку таблица может меняться в зависимости от ресурса, потребовалось использовать параметр `editable=True`.

Для выбора типа источника данных использовался класс `dcc.Dropdown`, для ввода - `dcc.Input`. [?]

Листинг 4: Форма добавления книги

```
app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
```

```

app.layout = html.Div([
    html.H1(
        children='Курсовой проект,',
        style={
            'textAlign': 'center',
            'color': '#7FDBFF'
        }
    ),

    html.Div(
        children='позволяющий сформировать список литературы для TeX',
        style={
            'textAlign': 'center',
            'color': '#7FDBFF'
        }
    ),

    html.Label('Для начала следует создать файл:'),
    html.Button(children = 'Создать файл', id='first_button', n_clicks = 0),
    html.Div(id="output0", children=''),

    html.Label('Введите название желаемой книги/документа/статьи etc.:'),
    dcc.Input(id = 'input1', type='text'),

    html.Label('Выберите тип источника:'),
    dcc.Dropdown(id = 'input2',
        options=[
            {'label': u'Книга', 'value': '1'},
            {'label': u'Нормативно-правовой документ', 'value': '2'},
            {'label': u'Нормативно-технический документ', 'value': '3'},
            {'label': u'Авторское свидетельство, патент', 'value': '4'},
            {'label': u'Информационные листки', 'value': '5'},
            {'label': u'Многотомные издания', 'value': '6'},
            {'label': u'Диссертации', 'value': '7'},
            {'label': u'Электронный ресурс локального доступа (CD)', 'value': '8'},
            {'label': u'Электронный ресурс удаленного доступа (Internet)', 'value': '9'},
            {'label': u'Статья из книги', 'value': '10'},
            {'label': u'Статья из журнала', 'value': '11'},
            {'label': u'Рецензия', 'value': '12'}
        ]
    ),

    html.Br(),
    html.Button(children = 'Добавить книгу', id='button', n_clicks = 0),
    html.Div(id="output", children=''),

    html.Label('После того, как все книги добавлены:'),
    html.Button(children = 'Получить файл', id='last_button', n_clicks = 0),
    html.Div(id="output2", children=''),
    html.Br(),
    html.Label('Тут можно посмотреть, что есть в БД:'),
    dcc.Dropdown(id = 'input3',
        options=[
            {'label': u'Книга', 'value': '1'},
            {'label': u'Нормативно-правовой документ', 'value': '2'},

```

```

{ 'label': u'Нормативно-технический документ', 'value': '3' },
{ 'label': u'Авторское свидетельство, патент', 'value': '4' },
{ 'label': u'Информационные листки', 'value': '5' },
{ 'label': u'Многотомные издания', 'value': '6' },
{ 'label': u'Диссертации', 'value': '7' },
{ 'label': u'Электронный ресурс локального доступа (CD)', 'value': '8' },
{ 'label': u'Электронный ресурс удаленного доступа (Internet)', 'value': '9' },
{ 'label': u'Статья из книги', 'value': '10' },
{ 'label': u'Статья из журнала', 'value': '11' },
{ 'label': u'Рецензия', 'value': '12' }
], value = '1'
),

dash_table.DataTable(
id='table',
columns=[{"name": i, "id": i} for i in columns_of_books],
data=[dict(zip(columns_of_books, d)) for d in books],
editable=True
)
])

```

Интерфейс приложения

На рис. 3 показан основной интерфейс приложения.

Курсовой проект,

позволяющий сформировать список литературы для TeX

Для начала следует создать файл:

создать файл

Введите название желаемой книги/документа/статьи etc.:

Выберите тип источника:

Select...

ДОБАВИТЬ КНИГУ

После того, как все книги добавлены:

получить файл

Тут можно посмотреть, что есть в БД:

Книга					X
id	name	author	year	extra_info	
1	Бухгалтерский учет	Краснова, Л.П.	2001	учебник для вузов	
2	Лесоводство	З.В. Ерохина	2000	учебное пособие к курсовому проектированию	

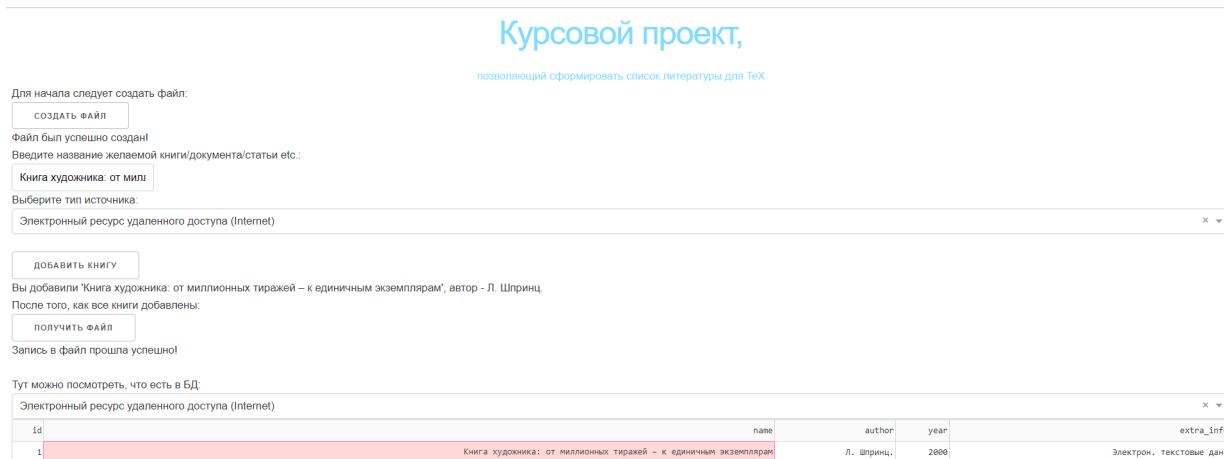
Рис. 3: Интерфейс приложения

Требования к аппаратуре

Пользовательский интерфейс представляет из себя одно страничное приложение для браузера, поэтому взаимодействие с сервисом может осуществляться только через браузер.

Тестирование

Протестируем программу, создав файл в web-приложении:



Курсовой проект,
позволяющий сформировать список литературы для TeX

Для начала следует создать файл:

создать файл

Файл был успешно создан!
Введите название желаемой книги/документа/статьи etc.:

Книга художника: от милл

Выберите тип источника:

Электронный ресурс удаленного доступа (Internet)

добавить книгу

Вы добавили 'Книга художника: от миллионов тиражей – к единичным экземплярам', автор - Л. Шпринц.
После того, как все книги добавлены:

получить файл

Запись в файл прошла успешно!

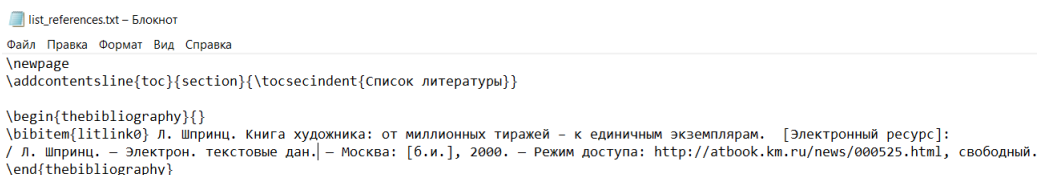
Тут можно посмотреть, что есть в БД:

Электронный ресурс удаленного доступа (Internet)

id	name	author	year	extra_info
1	Книга художника: от миллионов тиражей – к единичным экземплярам	Л. Шпринц.	2000	Электрон. текстовые дан.

Рис. 4: Демонстрация успешного создания файла

Тем временем, в папке с программой был действительно появился файл:



list_references.txt – Блокнот

Файл Правка Формат Вид Справка

```
\newpage
\addcontentsline{toc}{section}{\tocsecindent{список литературы}}

\begin{thebibliography}{}
\bibitem[litlink0]{Л. Шпринц. Книга художника: от миллионов тиражей – к единичным экземплярам. [Электронный ресурс]:
/ Л. Шпринц. – Электрон. текстовые дан. | – Москва: [б.и.], 2000. – Режим доступа: http://atbook.km.ru/news/000525.html, свободный.
\end{thebibliography}
```

Рис. 5: Созданный файл

Было проведено функциональное тестирование белого ящика, в ходе которого программа отработала правильно. Помимо этого, было проведено тестирование пользовательского интерфейса, все элементы интерфейса реагируют корректно.

Вывод

Было реализовано веб-приложение на языке Python. Программа полностью прошла функциональное тестирование и тестирование интерфейса.

Заключение

В результате проделанной работы:

- 1) были продуманы функции, которые должно было решать приложение;
- 2) проведен анализ инструментов, необходимых для проектирования и реализации задачи, в результате которого были выбраны такие инструменты, как MS-SQL, pyodbc, Dash;
- 3) разработана структура базы данных, состоящая из нескольких сущностей;
- 4) с помощью выбранных инструментов был реализован и протестирован web-интерфейс, позволяющий создавать файл литературы, а также просматривать содержимое библиотеки.

Литература

1. О правительственной комиссии по проведению административной реформы [Текст]: постановление Правительства РФ от 31 июля 2003 г. № 451 // Собрание законодательства. — 2003. — № 31. — Ст. 3150.
2. Краснова, Л.П. Бухгалтерский учет [Текст]: учебник для вузов / Краснова, Л.П., Н.Т. Шалашова, Н.М. Ярцева — Москва: Юристъ, 2001. — 550 с.
3. З.В. Ерохина Лесоводство [Текст]: учебное пособие к курсовому проектированию / З.В. Ерохина, Н.П. Гордина, Н.Г. Спицына, В.Г. Атрохин. — Красноярск: Изд-во СибГТУ, 2000. — 175 с.