МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: Хранилище документов в формате doc

Студенты гр. 6383 _____ Тимофеев Д.А.

_____ Михеева Е.Е.

Студентка гр. 6381 _____ Шевелева А.А.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2019

ЗАДАНИЕ

Студенты

Тимофеев Д.А.

Михеева Е.Е.

Шевелева А.А.
Группы 6383, 6381
Тема проекта: Хранилище документов в формате doc
Исходные данные:

Необходимо реализовать приложение для хранения документов в формате doc для СУБД MongoDB.
Содержание пояснительной записки:

«Содержание»

«Введение»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Заключение»

«Список использованных источников»

«Приложение»
Предполагаемый объем пояснительной записки:

Не менее 15 страниц.
Дата выдачи задания:
Дата сдачи реферата:
Дата защиты реферата:

Студенты гр.6383	Тимофеев Д.А.
	Михеева Е.Е.
Студентка гр.6381	Шевелева А.А.
Преподаватель	Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема хранения документов в формате doc для СУБД MongoDB. Приложение для хранения doc-документов позволит проверять лабораторные работы на принадлежность к определенному формату, искать работы по словам, фильтровать по размеру.

SUMMARY

As part of this course, it was supposed to develop any application in the team on one of the topics posed. The theme of document storage in doc format was chosen for the MongoDB DBMS. The application for storing doc-documents will allow you to check laboratory work for belonging to a specific format, search for work by words, filter by size.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – создать сервис для хранения документов в формате doc и для поиска документов по заданному слову, фильтрации документов по размеру и по наличию определенного формата.

Было решено разработать веб-приложение, которое позволит хранить в электронном виде все doc документов, позволяя при этом удобно с ними взаимодействовать.

Качественные требования к решению: требуется разработать приложение с использованием СУБД MongoDB.

1. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

1.1. Макеты пользовательского интерфейса

Стартовое окно представлено на рис. 1, окно дополнительной информации о doc-документе − на рис. 2, окно функций администратора − на рис. 3. Окна статистик представлены на рис. 4-6.

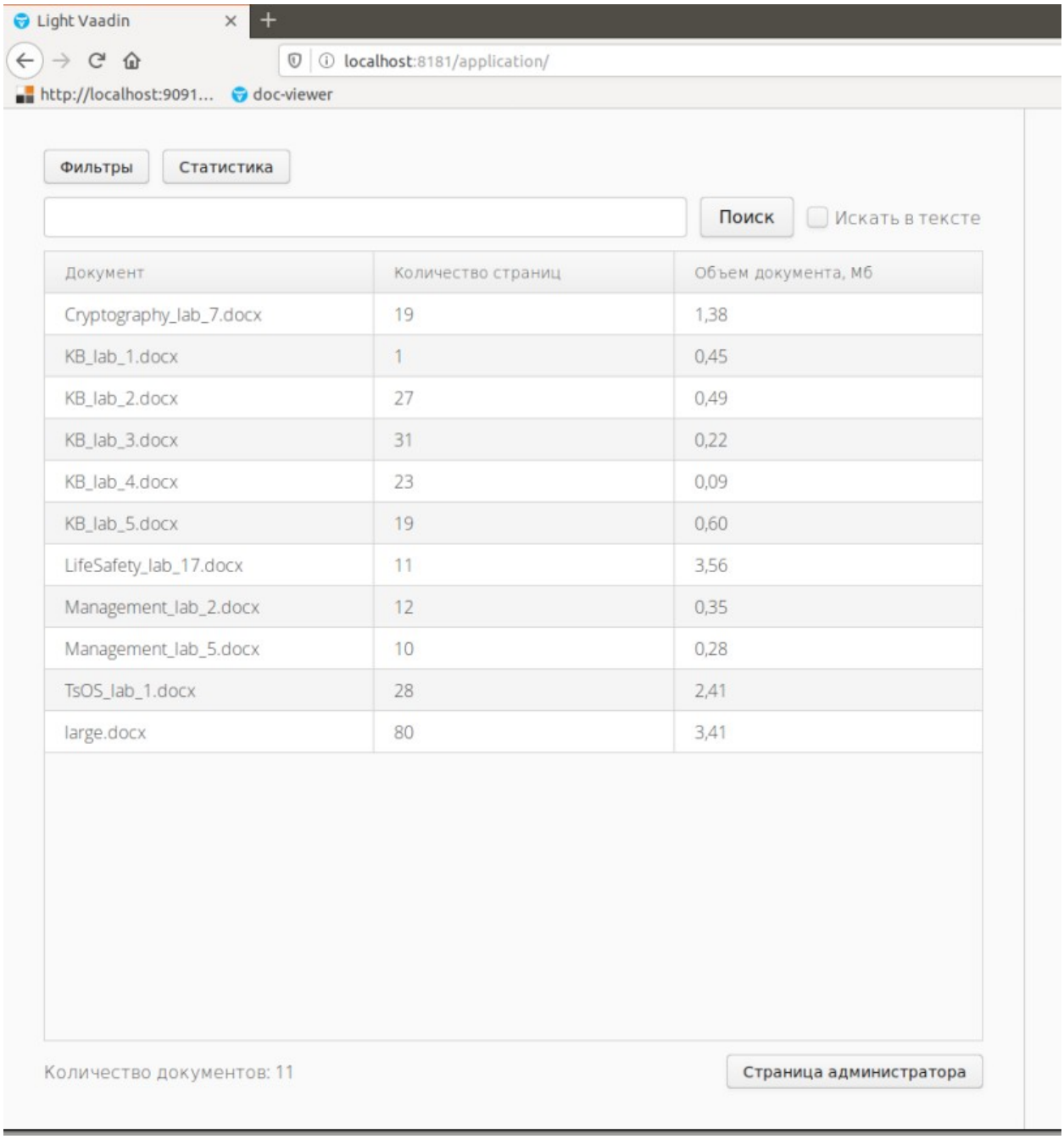

Рисунок 1 – Стартовое окно приложения

Рисунок 2 – Окно дополнительной информации о документе

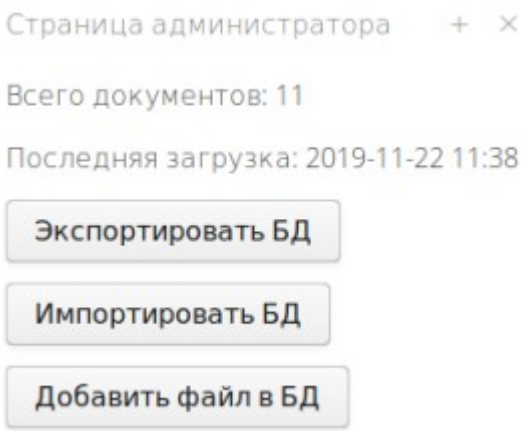

Рисунок 3 – Окно функций администратора

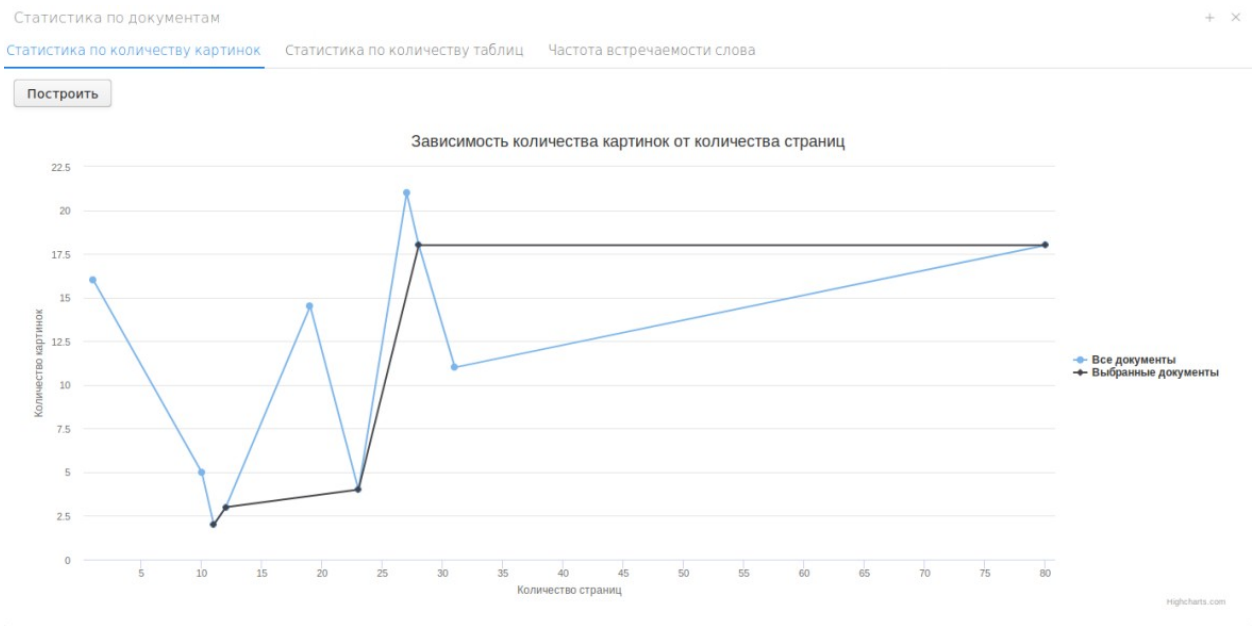

Рисунок 4 – Окно статистик: вкладка графика зависимости картинок от кол-ва страниц

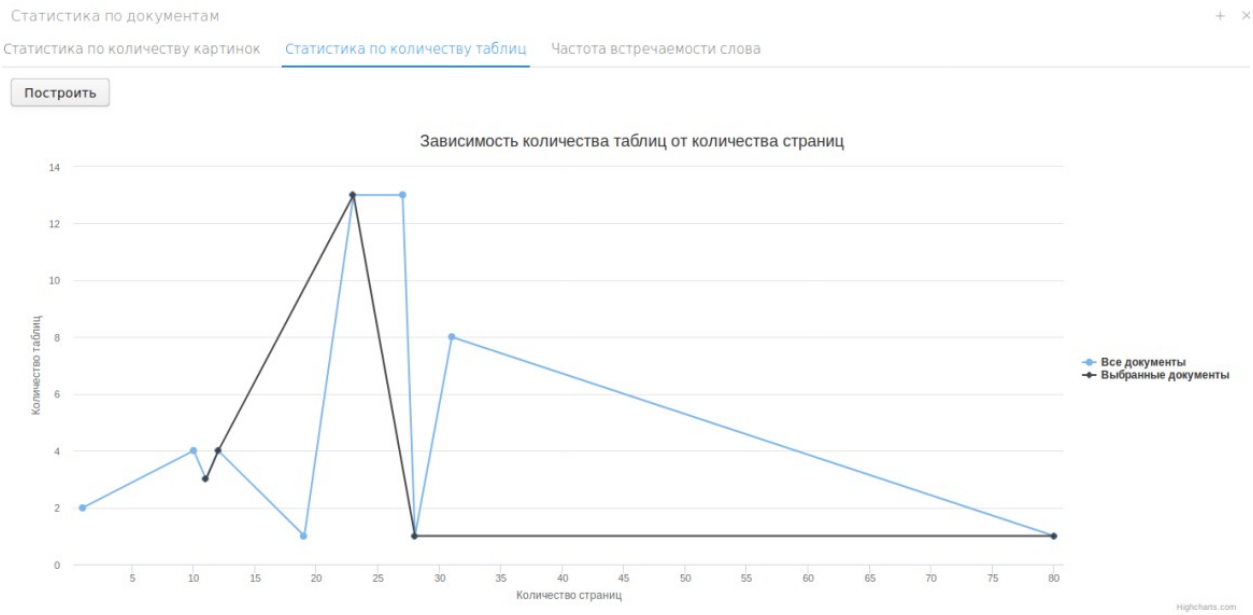

Рисунок 5 – Окно статистик: вкладка графика зависимости таблиц от кол-ва страниц

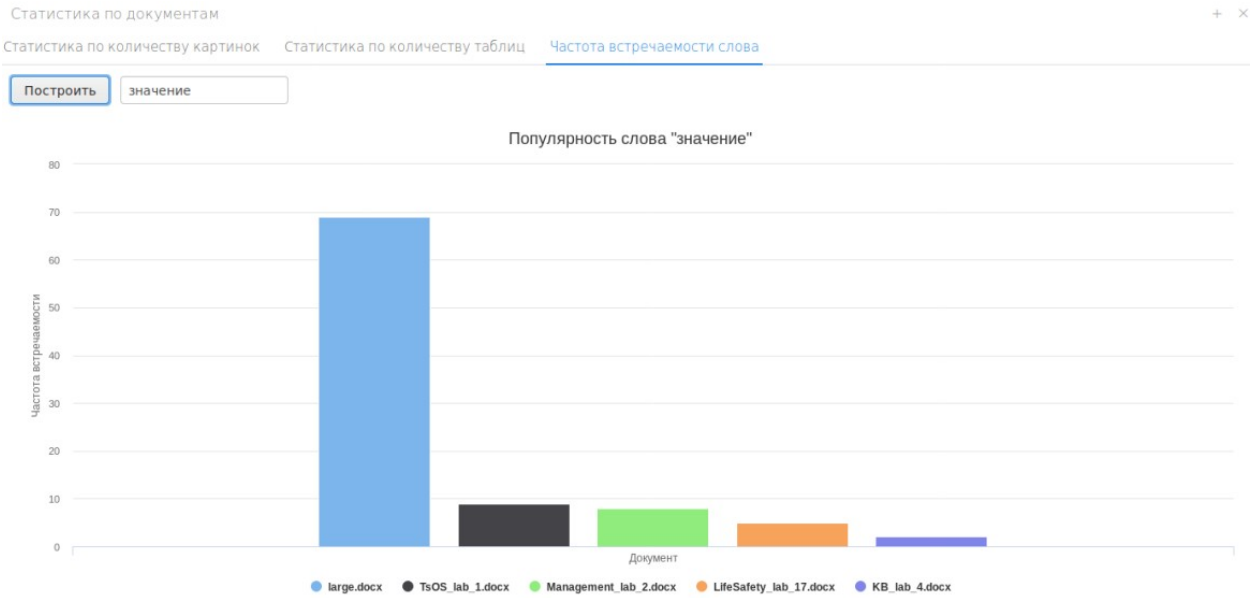

Рисунок 6 – Окно статистик: вкладка графика популярности слова в документах

1.2. Описание сценариев использования

1.2.1. Просмотр списка doc-документов

Основной сценарий:

1. Пользователь открывает страницу просмотра списка doc-документов.
2. Пользователь просматривает список doc-документов.

Результат:

Пользователь видит список всех doc-документов.

Альтернативный сценарий:

1. Пользователь осуществляет поиск doc-документов с помощью ввода строки в поле поиска.
2. Пользователь осуществляет поиск doc-документов по словам в самих документах с помощью ввода строки в поле поиска.

1.2.2. Фильтр списка doc-документов

Основной сценарий:

1. Пользователь нажимает кнопку "Фильтры".
2. Пользователь выбирает/отменяет свойство фильтрации "Документы с заданной структурой".
3. Пользователь выбирает/отменяет свойство фильтрации "Документы без заголовков".
4. Пользователь выбирает интервал количества страниц в документах.
5. Пользователь выбирает интервал объема документов.

Результат:

Пользователь видит список всех doc-документов, которые подходят под выбранные фильтры.

Альтернативный сценарий:

1. Пользователь отменяет применение фильтров.
2. Пользователь изменяет уже установленные ранее значения.

1.2.3. Функции администратора

Основной сценарий:

1. Пользователь нажимает кнопку "Страница администратора".

2. Пользователь выбирает действие администратора.
3. Пользователь нажимает на кнопку "Экспортировать БД".
4. Пользователь нажимает на кнопку "Импортировать БД".
5. Пользователь нажимает на кнопку "Добавить файл в БД".
6. Пользователь находит нужный документ в файловой системе компьютера.
7. Пользователь нажимает кнопку "Открыть".

Результат:

Пользователь либо добавляет документ в базу данных, либо экспортирует/импортирует все документы.

Альтернативный сценарий:

1. Пользователь отменяет выполнения действия администратора.
2. Пользователь отменяет выбор документа в файловой системе компьютера.

1.2.4. Просмотр информации по doc-документу

Основной сценарий:

1. Пользователь выбирает документ и нажимает на левую кнопку мыши.
2. Пользователь нажимает на правую кнопку мыши (открывает вспомогательное меню).
3. Пользователь нажимает на вкладку "Дополнительно".
4. Пользователь просматривает информацию о документе.

Результат:

Пользователь видит информацию по выбранному doc-документу.

Альтернативный сценарий:

Пользователь закрывает вспомогательное меню.

1.2.5. Просмотр диаграмм статистики по документу

Основной сценарий:

1. Пользователь выбирает документ и нажимает на левую кнопку мыши.
2. Пользователь нажимает на кнопку "Статистика".
3. Пользователь переходит на вкладку "Статистика по количеству картинок".
4. Пользователь переходит на вкладку "Статистика по количеству таблиц".
5. Пользователь переходит на вкладку "Частота встречаемости слова".

Результат:

Пользователь видит диаграммы по выбранному doc-документу.

Альтернативный сценарий:

1. Пользователь отменяет просмотр диаграмм статистик.
2. Пользователь меняет вид диаграммы, путем нажатия на данные легенды диаграммы.
3. Пользователь нажимает на кнопку "Построить" для обновления диаграммы.

1.2.6. Просмотр частоты встречаемости слова

Основной сценарий:

1. Пользователь выбирает вкладку "Частота встречаемости слова" (см. пункт "Просмотр диаграмм статистики по документу").
2. Пользователь вводит слово в окно построения.
3. Пользователь нажимает на кнопку "Построить".

Результат:

Пользователь видит диаграмму частоты введенного слова в выбранном документе относительно остальных документов.

Альтернативный сценарий:

Пользователь выбирает построение диаграммы "По всем документам".

2. МОДЕЛЬ ДАННЫХ

2.1. NoSQL модель данных (MongoDB)

2.1.1. Графическое представление

Графическое представление модели данных MongoDB показано на рис. 7.

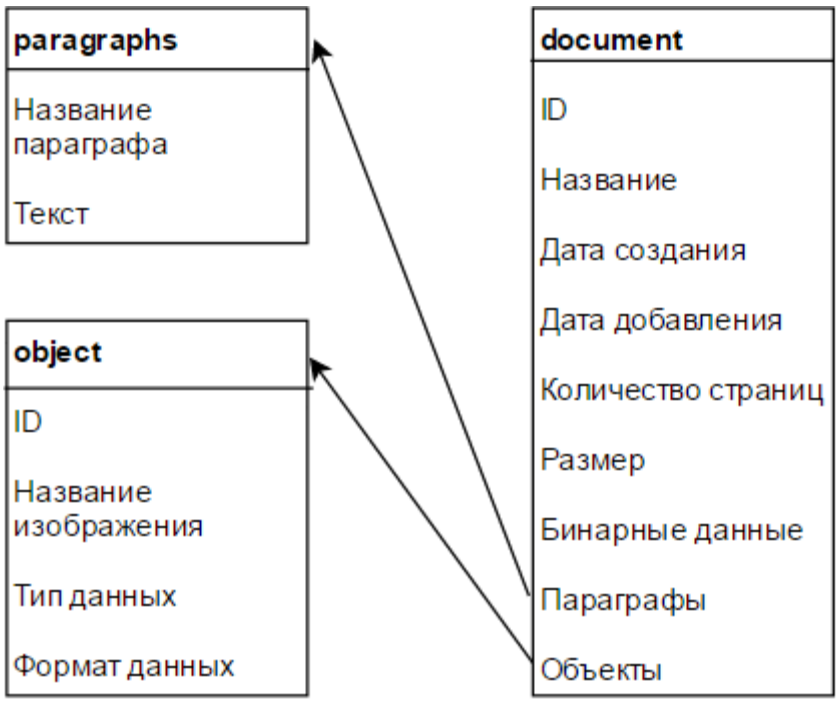

Рисунок 7 − Графическое представление модели данных MongoDB

2.1.2. Описание назначений коллекций, типов данных и сущностей

В качестве СУБД используется MongoDB, в которой содержится одна коллекция document, хранящая данные о документах, их названиях, даты создания и добавления, размере, количестве страниц, параграфах, изображениях и таблицах. Каждый документ коллекции document содержит следующие поля:

_id − уникальный идентификатор документа. Тип - ObjectID. V = 12b.

name − Название документа. Тип - string. V = 2b*N, где N − средняя длина ключа задачи. На основе загруженных данных установлено, что в среднем N = 25. Тогда V = 50b.

createDate − дата создания. Тип − date. V = 8b.

addDate - дата добавления. Тип − date. V = 8b.

pageCount − количество страниц в документе. Тип − int. V = 4b*2 = 8b.

size − размер документа. Тип − string. V = 4b*3 = 12b.

paragraphs − параграфы документа. Тип − array. Содержит массив документов с полями:

name − наименование параграфа. Тип − string. V = 2b*4 = 8b.

TEXT − это текст относящийся к этому параграфу Тип − binary data. V = 1140b.

binData − бинарное представление файла (путь к файлу). Тип − string. V = 2b*11 = 22b

Objects − объекты документа (таблицы, изображения). Тип − array. Содержит массив документов с полями:

_id − уникальный идентификатор документа. Тип − ObjectID. V = 12b.

TYPE − указание типа объекта. Тип − string. V = 2b*5 = 10b.

name − название объекта. Тип − string. V = 2b*35 = 70b.

format − формат объекта (только для изображений). Тип − string. V = 2b*4 = 8b.

2.1.3. Расчет объема

"Чистый" объем: Объем документа в коллекции document − 1056b, при условии, что в массиве paragraphs 3 элемента, а в массиве objects − 4 изображения и 1 таблица. Объем всей БД V = 4056b * M, где M − количество записей о документах в БД.

Фактический объем: Объем документа в коллекции document − 4232b, при условии, что в массиве paragraphs 3 элемента, а в массиве objects − 4 изображения и 1 таблица. Объем всей БД V = 4232b * M, где M − количество записей о документах в БД.

Избыточность модели: 4232b / 4056b.

2.1.4. Примеры запросов

Количество параграфов у данного документа:

```
db.docs.find({"_id": "1"}).map(a=>a.pars.length)
```

Поиск документов, у которых есть параграфы "Введение" и "Заключение":

```
db.docs.find({'pars':[{'name':'vved'},
{'name':'zakl'}]})
```

2.2. SQL модель данных

2.2.1. Графическое представление

Графическое представление модели данных SQL показано на рис. 8.

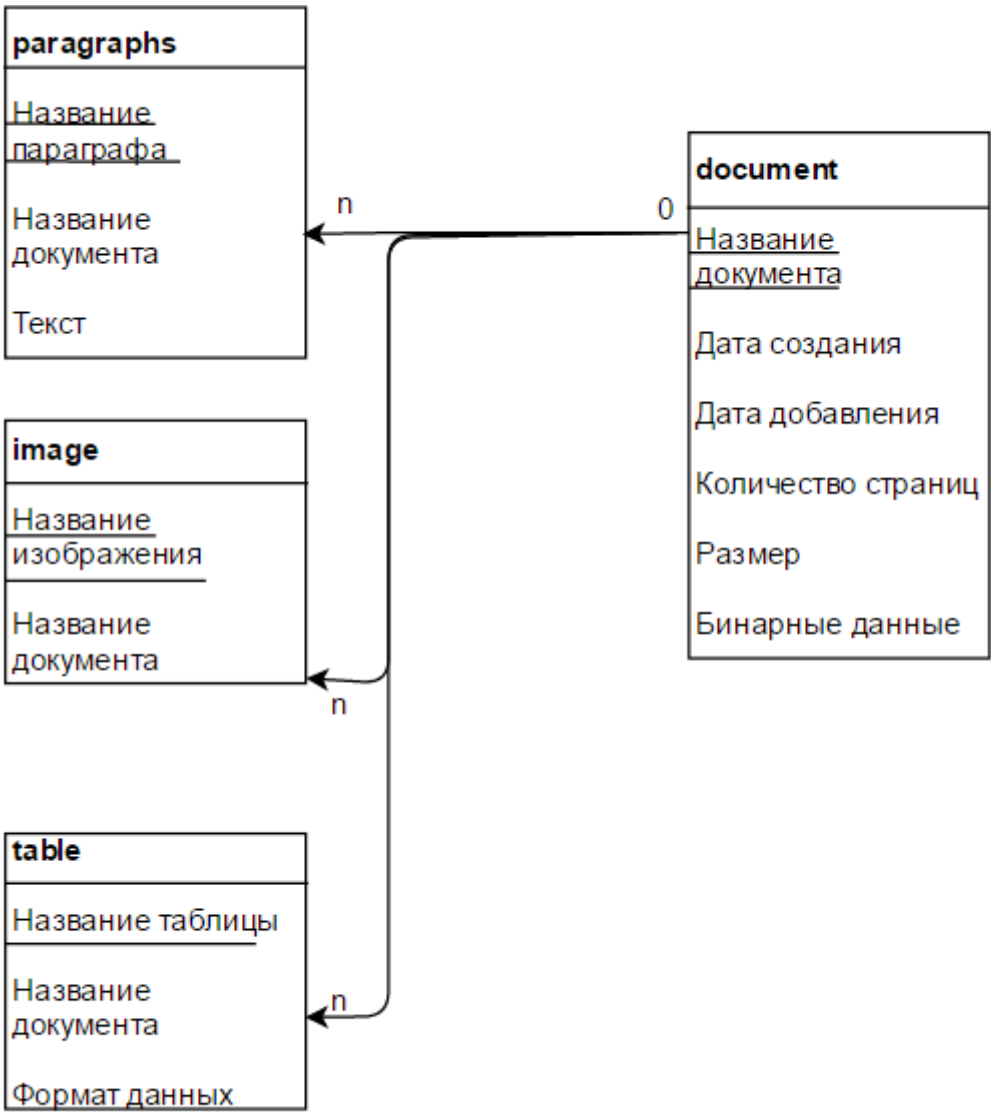

Рисунок 8 − Графическое представление модели данных SQL

2.2.2. Описание назначений коллекций, типов данных и сущностей

В качестве реляционной СУБД использована MySQL, в которой создано 4 таблицы: document − хранящая данные о документах; image − содержащая сведения об изображениях; table − содержащая сведения о таблицах; paragraphs − хранящая данные о параграфах.

- Таблица document содержит следующие поля:

  name − Название документа . Тип − varchar. V = 2b*25 = 50b.

  createDate − дата создания. Тип − date. V = 8b.

  addDate − дата добавления. Тип − date. V = 8b.

  pageCount − количество страниц в документе. Тип − int. V = 4b*2 = 8b.

  size − размер документа. Тип − string. V = 4b*3 = 12b.

  binData − бинарное представление файла (путь к файлу). Тип − varchar. V = 2b*11 = 22b.

- Таблица image содержит следующие поля:

  name_doc − Название документа . Тип − varchar. V = 2b*25 = 50b.

  name − Название изображения. Тип − varchar. V = 2b*35 = 70b.

  format − Формат изображения. Тип − varchar. V = 2b*4 = 8b.

- Таблица table содержит следующие поля:

  name_doc − Название документа . Тип − varchar. V = 2b*25 = 50b.

  name − Название таблицы. Тип − varchar. V = 2b*35 = 70b.

- Таблица paragraphs содержит следующие поля:

  name_doc − Название документа . Тип − varchar. V = 2b*25 = 50b.

  name − наименование параграфа. Тип − varchar. V = 2b*4 = 8b.

  TEXT − это текст относящийся к этому параграфу Тип − TINYINT.V = 1140b.

2.2.3. Расчет объема

"Чистый"объем: объем записи в таблице document в среднем составляет 108b, в таблице image − 128b, в таблице table − 120b, в таблице paragraphs

−1198b. Объем всей БД V = 108b * M + 3 * 128b * M + 1 * 120b * M + 3 * 1198b * M = 4206b * M, где M - количество записей о документах в БД.

Фактический объем: объем записи в таблице document в среднем составляет 108b, в таблице image − 132b, в таблице table − 128b, в таблице paragraphs − 1246b. Объем всей БД V = 108b * M + 3 * 132b * M + 1 * 128b * M + 3 * 1246b * M = 4370b * M, где M - количество записей о документах в БД.

Избыточность модели: 4370b / 4206b.

2.2.4. Примеры запросов

Количество параграфов у данного документа:

```
select count(*) from paragraphs where docId=1;
```

Поиск документов, у которых есть параграфы "Введение" и "Заключение":

```
select docId from paragraphs inner join (select distinct docId as di from paragraphs where name="vved") as tmp on di=docId where name="zakl";
```

2.3. Сравнение MongoDB и SQL моделей данных

Таким образом, получается примерно одинаковый объем базы данных в MongoDB и в SQL, так как они имеют похожую структуру. В SQL были созданы отдельные таблицы изображений, таблиц, параграфов, в MongoDB были созданы массивы параграфов и объектов (таблицы и изображения) внутри коллекции документов.

В силу того что в MongoDB на структуру документа накладывается меньше ограничений, чем в реляционной базе данных, в случае добавления ранее не предусмотренных объектов (таких как формулы) в MongoDB нужно просто добавить запись с соответствующим значением TYPE для элемента массива objects. В реляционной базе данных для каждого нового типа

объекта потребуется создать новую таблицу, аналогично тому как мы создали отдельные таблицы для хранения изображений и таблиц.

При этом время работы двух запросов в Mongodb составило 4ms, в то время как на выполнение запроса в SQL потребовалось 16ms (в базе с 10 документами).

Исходя из всего выше сказанного, можно сделать вывод о том, что MongoDB больше подходит для задач подобного рода.

3. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

3.1. Краткое описание

Для упрощения процесса разработки было принято решение разделить приложение на front-end и back-end части.

Front-end часть реализована с использованием Vaadin фреймворка и js библиотеки highcharts.

Back-end часть приложения реализована с использованием Java. Для реализации функции предпросмотра doc-файлов в браузере была использована программа unoconv для предварительной конвертации файла в формат pdf.

Поскольку приложение требует установки стороннего программного обеспечения: mongodb, java и unoconv − было решено для упрощения развертывания приложения использовать docker.

Ссылка на приложение доступна в разделе "Список использованных источников": [1]

3.2. Схема экранов приложения

Схема экранов приложения представлена на рис. 9.

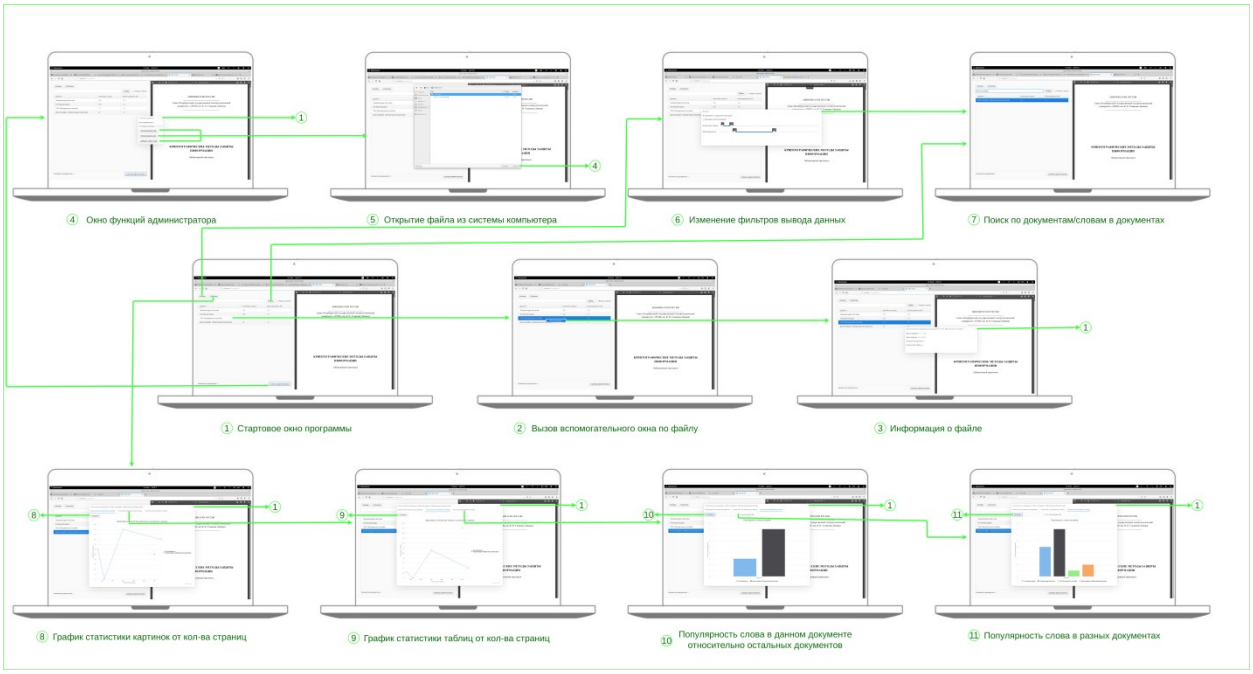

Рисунок 9 – Схема экранов приложения

3.3. Использованные технологии

БД: MongoDB

Backend: Java

Frontend: Vaadin, Highcharts

ЗАКЛЮЧЕНИЕ

В ходе работы было разработано приложение, позволяющее пользователю осуществлять поиск по словам документов в формате doc, хранящихся в базе данных приложения. Так же, была реализована возможность добавлять документы, просматривать текст документов, и фильтровать документы по объему, количеству страниц и определенному формату документа. В приложение была добавлена статистика по таблицам и изображениям в зависимости от страниц и статистика по частоте встречаемости слова. В учебных целях были добавлены import и export датасетов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Репозиторий проекта: https://github.com/moevm/nosql2h19-doc
2. Документация MongoDB: https://docs.mongodb.com
3. Документация Vaadin: https://vaadin.com
4. Документация Highcharts: https://www.highcharts.com/

ПРИЛОЖЕНИЕ А
ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ

Для использования приложения требуется наличие docker.

Установка приложения выполняется с помощью команды:

```
docker pull dimotim/docdb
```

Запуск приложения выполняется с помощью команды:

```
docker run -p 8181:8181 -i -t dimotim/docdb
```

После запуска приложение доступно по адресу:

```
http://localhost:8181/application/
```