# МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ**

**по дисциплине «Введение в нереляционные базы данных» Тема: Система хранения, обработки и анализа ROSbag-наборов данных**

Студенты гр. 6381 Фиалковский М.С.

Афийчук И.И.

Сергухин В.Ю.

Преподаватель Заславский М.М.

Санкт-Петербург 2019

# ЗАДАНИЕ

Студенты Фиалковский М.С., Афийчук И.И., Сергухин В.Ю..

Группа 6381

Тема проекта: Система хранения, обработки и анализа ROSbag-наборов данных.

Исходные данные:

Необходимо реализовать приложение, использующее СУБД MongoDB

Содержание пояснительной записки:

«Содержание», «Введение», «Качественные требования к решению»,

«Сценарий использования», «Модель данных», «Разработанное приложение»,

«Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки: Не менее 10 страниц.

Дата выдачи задания: Дата сдачи ИДЗ:

Дата защиты ИДЗ:

# АННОТАЦИЯ

В рамках данного курса необходимо было реализовать приложение на одну из поставленных тем. Была выбрана тема для создания приложения, которое хранит ROSbag – наборы данных. В приложении должна осуществляться функция импорта/экспорта данных.

# SUMMARY

As part of this course, it was necessary to implement the application on one of the topics posed. A theme was chosen to create an application that stores ROSbag - datasets. The application should carry out the function of import / export of data.

# СОДЕРЖАНИЕ

1. [КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ 5](#_bookmark0)
2. [СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ 5](#_bookmark1)
   1. [Макеты пользовательского интерфейса 5](#_bookmark2)
   2. [Описание сценариев использования](#_bookmark3) 9
      1. [Сценарий использования](#_bookmark4)  9
      2. [Сценарий использования](#_bookmark5)  9
      3. [Сценарий использования](#_bookmark6)  10
3. [МОДЕЛЬ ДАННЫХ 11](#_bookmark7)
   1. [NoSQL модель данных 11](#_bookmark8)
      1. [Графическое представление 11](#_bookmark9)
      2. [Подробное описание и расчёт объема 11](#_bookmark10)
      3. [Примеры запросов 13](#_bookmark11)
   2. [SQL модель данных 14](#_bookmark12)
      1. [Графическое представление 14](#_bookmark13)
      2. [Подробное описание и расчёт объема 14](#_bookmark14)
      3. [Примеры запросов 15](#_bookmark15)
   3. [Сравнение NoSQL и SQL моделей данных 1](#_bookmark16)5
4. [РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ 17](#_bookmark17)
   1. [Краткое описание 17](#_bookmark18)
   2. [Схема экранов приложения 17](#_bookmark19)
   3. [Использованные технологии 18](#_bookmark20)
   4. [Ссылка на приложение 18](#_bookmark21)

[ЗАКЛЮЧЕНИЕ 19](#_bookmark22)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#_bookmark23)

ПРИЛОЖЕНИЕ… 21

**ВВЕДЕНИЕ**

Цель работы – создать приложение, которое позволяет хранить, обрабатывать и анализировать ROSbag – наборы данных.

Было решено разработать веб-приложение, представляющее из себя, файловый менеджер, позволяющий отсортировать файлы с помощью фильтров, а также просматривать существующие теги внутри файлов и переменные, собранные с сенсоров.

# КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется реализовать веб-приложение, использующее СУБД MongoDB.

# СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

# Макеты пользовательского интерфейса

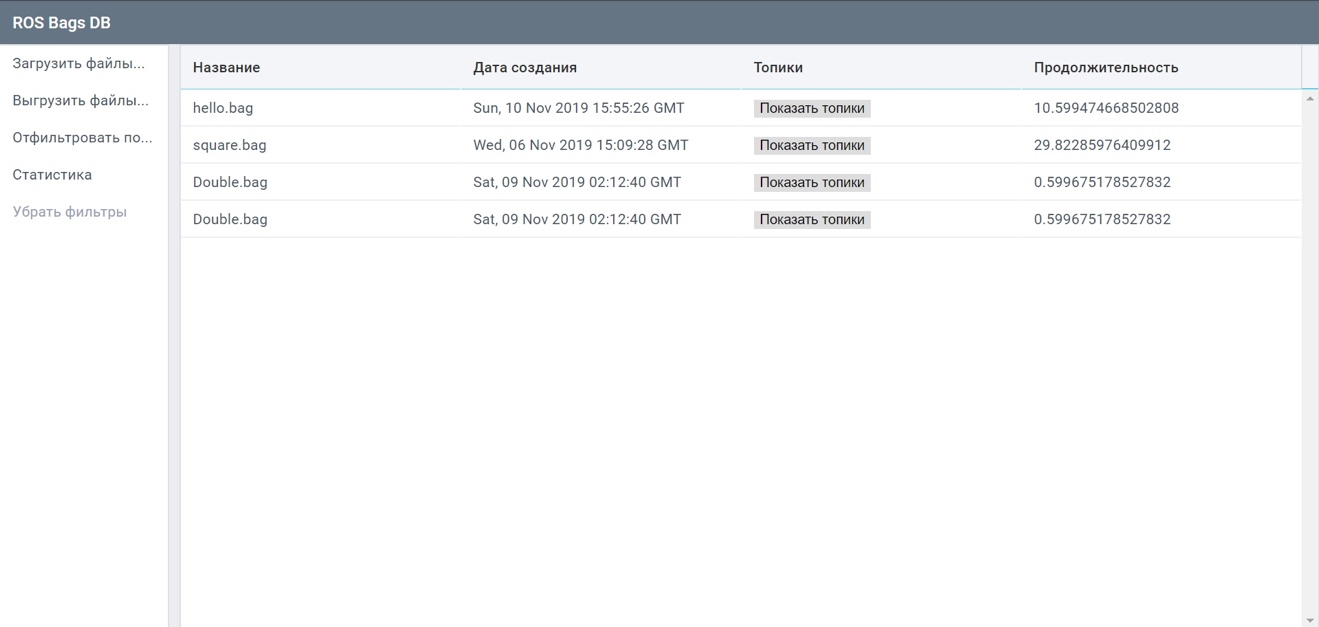
****

Рисунок 1 — Окно страницы «Просмотр»

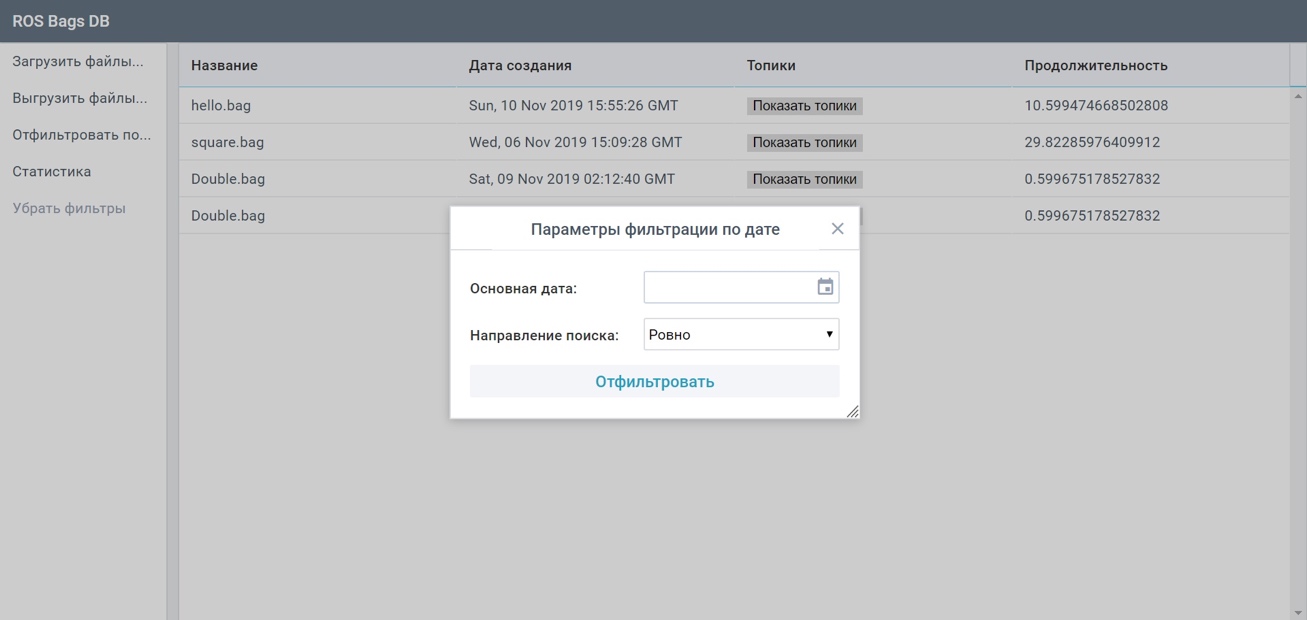


Рисунок 2 — Интерфейс способов фильтрации

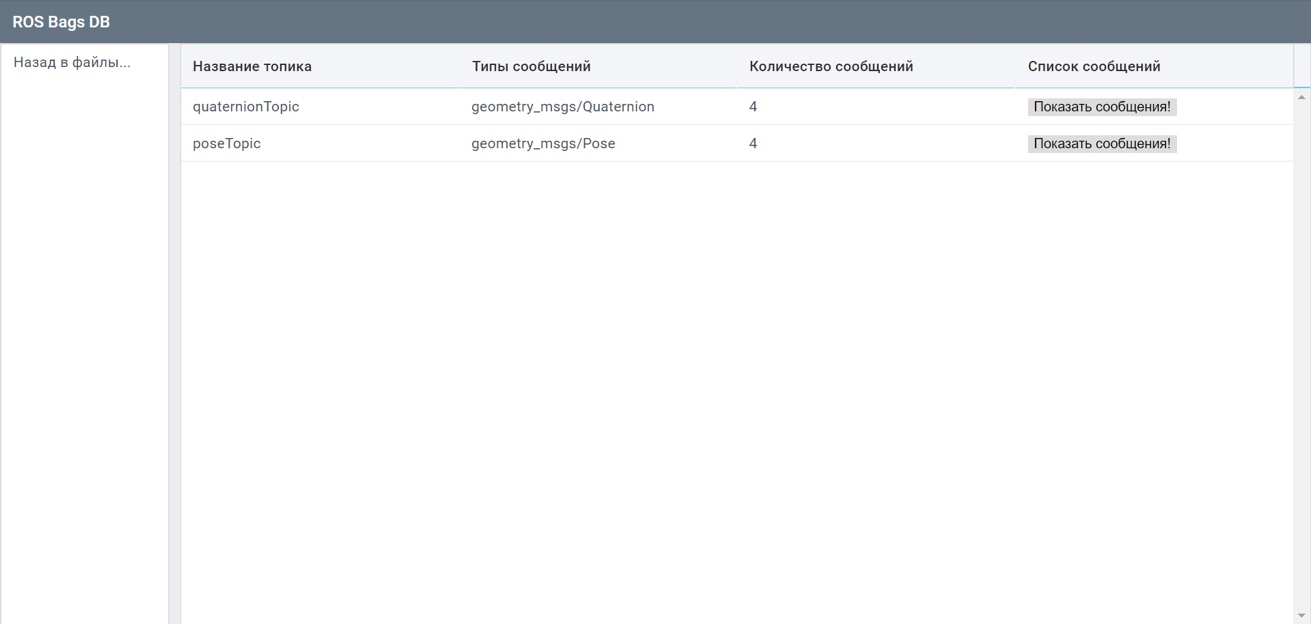


Рисунок 3 — Окно страницы «Топики»

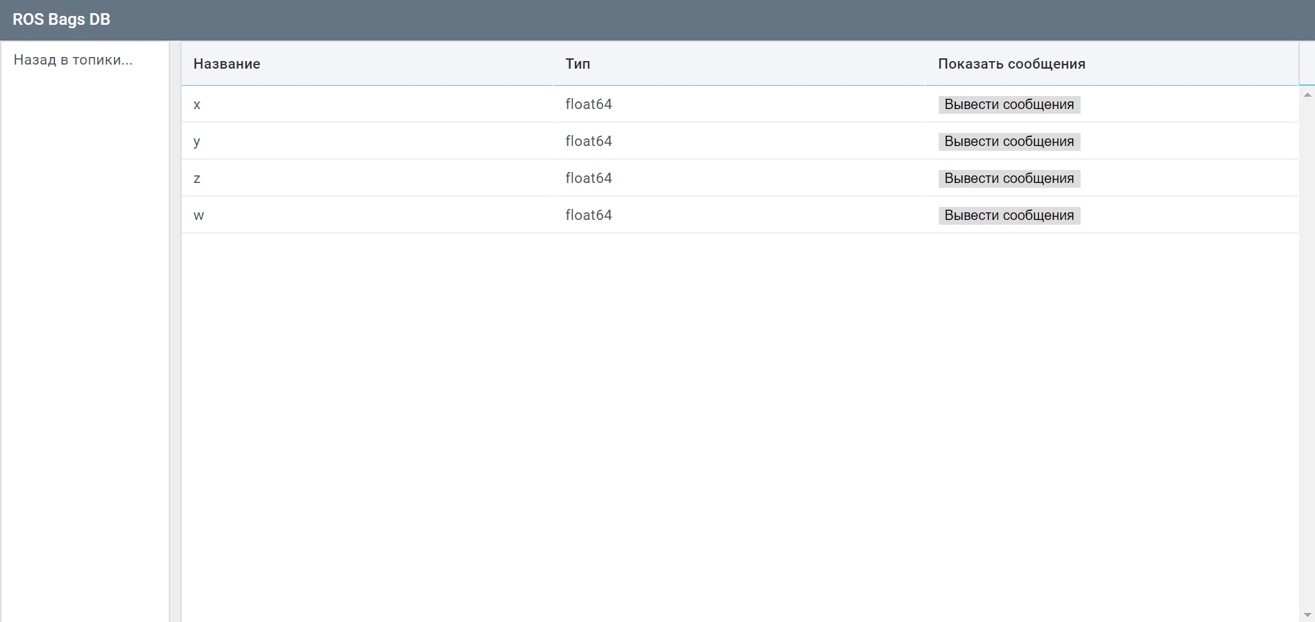


Рисунок 4 — Окно страницы «Сообщения»

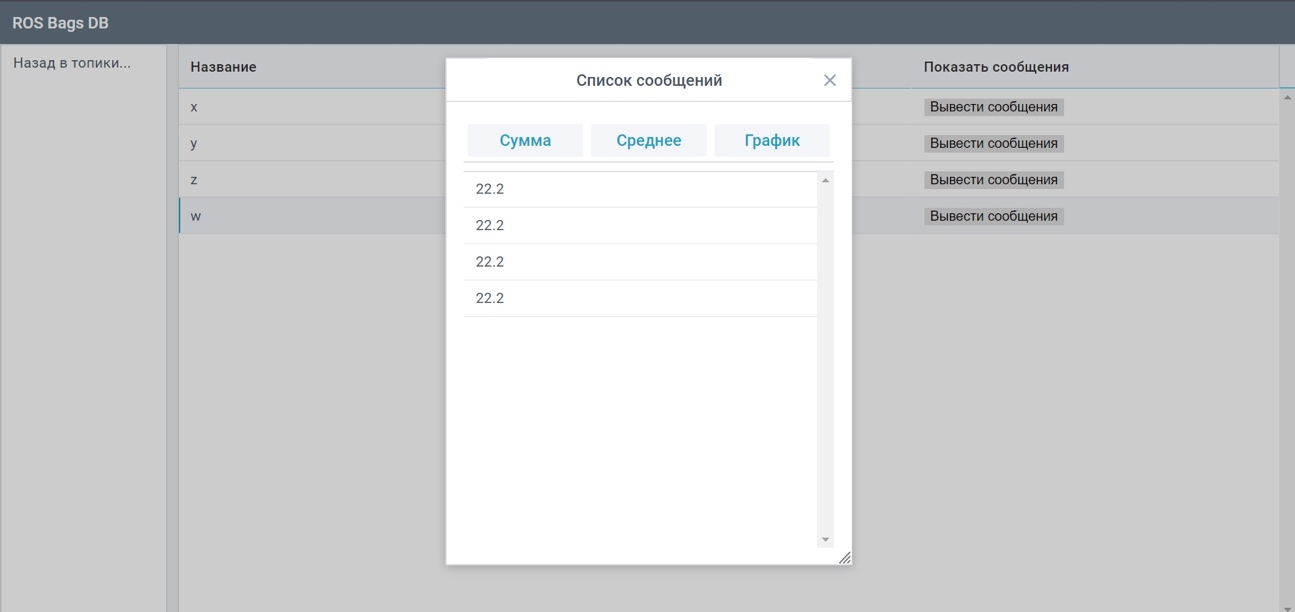


Рисунок 5 — Окно страницы «Список сообщений»

При использовании приложения пользователь делает выбор страницы: «Просмотр», «Топики», «Сообщения», «Список сообщений». Результат: переход на страницы «Просмотр», «Топики», «Сообщения» и «Список сообщений» соответственно.

**2.2.1. Сценарий использования страницы «Просмотр»**

1) Пользователь совершает импорт таблицы. Результат: загруженная страница.

2) Пользователь совершает экпорт таблицы. Результат: выгруженная страница.

3) Пользователь отбирает данные по определенным критерий с помощью окна «отфильтровать по …», внутри которого выбирает фильтр «Дата». Результат: программа оставляет в окне просмотра соответствующие файлы, созданные в установленных фильтром промежутках.

4) Пользователь отбирает данные по определенным критерий с помощью окна «отфильтровать по …», внутри которого выбирает фильтр «Топики». Результат: программа оставляет в окне просмотра соответствующие файлы, содержащие внутри себя определенный топик.

5) Пользователь отбирает данные по определенным критерий с помощью окна «отфильтровать по …», внутри которого выбирает фильтр «Времени записи». Результат: программа оставляет в окне просмотра соответствующие файлы, продолжительностью, установленной пользователем в фильтре.

6) Пользователь нажимает на поле показать «Показать топики». Результат: Пользователь переходит на страницу «Топики».

**2.2.2. Сценарий использования страницы «Топики»**

1) Пользователь нажимает на поле показать «Показать Сообщения». Результат: Пользователь переходит на страницу «Сообщения».

2) Пользователь нажимает на кнопку «Назад в файлы». Результат: Программа возвращается на страницу «Просмотр».

**2.2.3. Сценарий использования страницы «Сообщения»**

1) Пользователь нажимает на поле показать «Вывести сообщения». Результат: Программа выводит окно со списком сообщений.

2) Пользователь нажимает на кнопку «Назад в топики». Результат: Программа возвращается на страницу «Топики».

**2.2.4. Сценарий использования страницы «Список сообщений»**

1) Пользователь нажимает на кнопку закрытия окна. Результат: Программа возвращается на страницу «Сообщения».

2) Пользователь нажимает на кнопку «Сумма». Результат: программа выдаст сумму всех значений, хранящихся внутри сообщения.

3) Пользователь нажимает на кнопку «Среднее». Результат: программа выдаст среднее значение всех значений.

4) Пользователь нажимает на кнопку «График». Результат: программа выдаст график, построенный на основе значений, находящихся внутри сообщения.

# МОДЕЛЬ ДАННЫХ

# NoSQL модель данных

# Подробное описание и расчёт объема

**Оценка удельного объема информации в модели и скорость её роста**

Строки в MongoDB хранятся в формате UTF-8, поэтому один символ строки будет занимать 4 байта, а среднее количество символов в строке возьмём за 10. Целое - 4 байта, дробное - 8 байт. Пусть количество топиков в одном файле равно Topics, количество типов сообщений в топике - MT, а количество сообщений - Msgs.

Тогда общий чистый объем БД можно оценить следующим образом: N \* (12 + 40 + 64 + 8 + T \* (40 + 4 + 40 + MT \* (40 + 40 + 8Msgs))) = (((80 + 8 \* Msgs) \* MT + 84) \* T + 124) \* N, где N - количество файлов в базе данных. Рост будет происходить линейным образом.

Приняв средние значения Topics = 4, MT = 5, Msgs = 1000 получим зависимость: size = 162060 \* N = 160 Кб \* N.

**Избыточность модели**

Сама по себе модель не несёт в себе избыточности, т.к было использовано естественное представление первичных данных. Но существует возможность выделения отдельной сущности "тип сообщений" с её последующим включением по ссылке. В среднем такое улучшение улучшит оценку на 70 байт на один файл, т.е. realSize - idealSize = 70N. Итого для одного файла: 162060 / (162060 - 70)= 1,0004.

Таким образом, на каждую запись в среднем приходится 70 потенциально дублирующихся байт данных, и реальный объем БД потенциально больше идеального размера в 1.0004 раза больше.

Можно сделать вывод, что данное улучшение неоптимально.

# Примеры запросов



Рисунок 6 — Запрос на число файлов, помещенных в БД за период с Date 1

по Date 2



Рисунок 7 — Запрос на сумму значений в файле fileName в топике topicName в сообщения с именем 'name'

# SQL модель данных

# Графическое представление

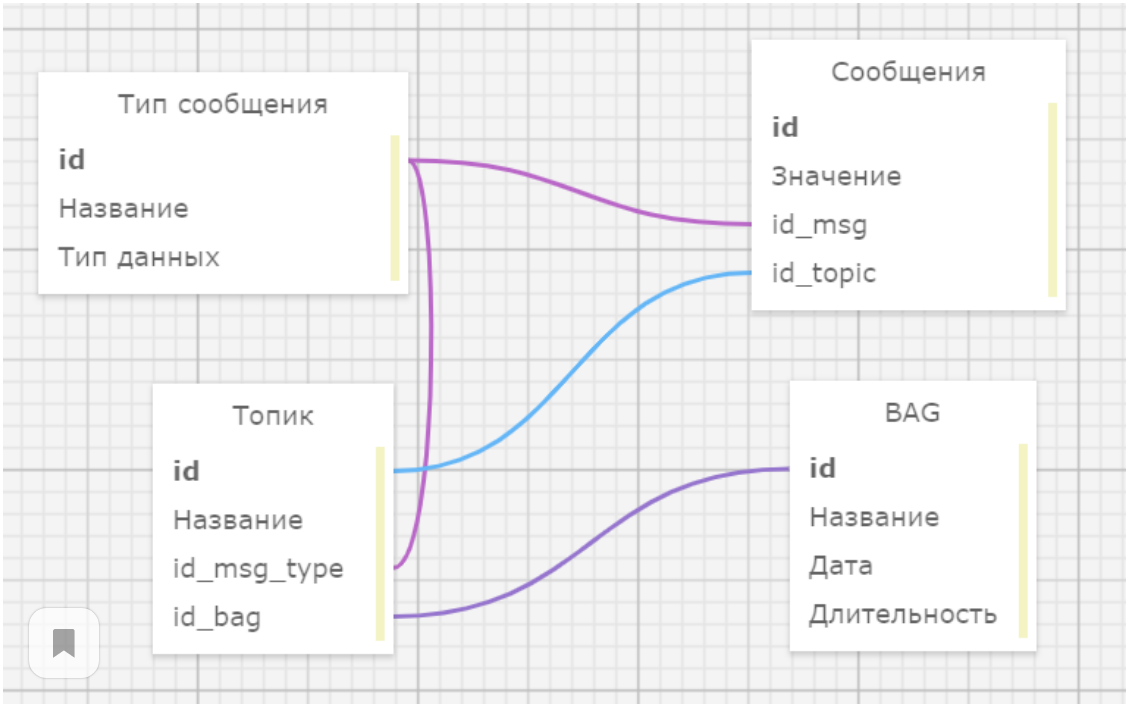


Рисунок 8 — Графическое представление SQL модели

# Подробное описание и расчёт объема

**Описание назначения коллекций, типов данных и сущностей**

Таблица "Тип сообщения" хранит в себе данные о типе сообщения и названии сообщения.

Таблица "Сообщение" хранит значение, внешний ключ о типе сообщения и внешний ключ и топике, которому принадлежит.

Таблица "Топик" хранит название топика, имея внешний ключ о Bag, которому принадлежит и внешний ключ на тип хранимых сообщений.

Таблица Bag хранит название файла, его размер и длительность.

Также при хранении в топике более одного типа сообщений требуется ещё одна таблица для реализации связи один ко многим.

**Оценка удельного объема информации в модели и скорость её роста**

Будем также считать, что символ строки будет занимать 4 байта, а среднее количество символов в строке возьмём за 10. Целое - 4 байта, дробное - 8 байт. Пусть количество топиков в одном файле равно Topics, количество типов сообщений в топике - MT, а количество сообщений - Msgs.

Тогда общий чистый объем БД можно оценить следующим образом: N \* ((Msgs \* (4\*3 + 8) + 3 \* 4 + 40 + 40 + 40) \* Topics \* MT) = ((20Msgs + 132) \* Topcs \* MT)N, где N - количество файлов в базе данных. Рост будет происходить линейным образом.

\**Приняв средние значения Topics = 4, MT = 5, Msgs = 1000 получим зависимость: size = 402640 \* N = 400 Кб \* N.*

**Избыточность модели**

Используя полученные формулы чистого и фактического объемов, вычислим значения избыточности:

SQLsize / pureSize = 400 Кб \* N / 140 Кб \* N = 2.86

SQLsize - pureSize = 400 Кб \* N - 140 Кб \* N = 242N

Таким образом, на каждую запись приходится в среднем 242 Кб дублирующихся данных. Фактический объем БД больше чистого объёма данных в 2.86 раз.

Таким образом, SQL-модель более избыточна вследствие хранения идентификаторов и внешних связей по ключам в каждой таблице. Данная реализация на SQL проигрывает NoSQL реализации по избыточности.

# Примеры запросов

# 

# Рисунок 9 — Запрос на число файлов, помещенных в БД за период с Date1 по Date2

# 

# Рисунок 10 — Запрос на сумму значений в файле fileName в топике topicName в сообщения с именем 'name'

# 

# Сравнение NoSQL и SQL моделей данных

NoSQL требует заметно меньше памяти - в среднем 160N против 400N у SQL-решения.

NoSQL также выигрывает по качеству запросов: большинство запросов выполняется внутри одного документа (таблицы), остальные запросы могут просматривать все документы. В это же время SQL-решение потребовало бы ещё дополнительных запросов к нескольким таблицам по внешним ключам, что может уменьшить сравнительное время отклика у двух систем.

# РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

**Краткое описание**

Для упрощения процесса разработки было принято решение разделить приложение на frontend и backend части.

Frontend-часть реализована с использованием Webix [3]. Webix — это JavaScript фреймворк, с помощью которого можно создавать десктопные и мобильные веб-приложения с отзывчивым дизайном. Фреймворк доступен под двумя лицензиями: GNU GPLv3 и коммерческой.

Backend-часть приложения реализована с использованием Flask[4]. Flask — фреймворк для создания веб-приложений на языке программирования Python, Относится к категории так называемых микрофреймворков — минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности.

**Схема экранов приложения**

****

Рисунок 11 — Схема экранов приложения

# Использованные технологии

* ROS + python-API – чтение и создание файлов формата rosbag
* MongoDB + PyMongo – база данных для хранения и обработки наборов данных (Python)
* Flask – python-фреймворк для бэкенда
* Webix – фронтенд js библиотека

# Ссылка на приложение

Ссылка на приложение доступна в разделе «Список использованных источников»[5].

# ЗАКЛЮЧЕНИЕ

В ходе выполнения задания было разработано приложение, в котором можно смотреть информацию содержащуюся в ROS-файлах. В приложении есть функции импорта/экспорта данных. Кроме того, пользователь может просмотреть различную статистику, выбирая необходимые фильтры. Таким образом, цель, поставленная перед началом работы, достигнута.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

# 1)Документация MongoDB: <https://docs.mongodb.com/manual/>

# 2)Документация ROS + python-API: http://wiki.ros.org/ROS/API

# 3)Документация Flask: https://flask.palletsprojects.com/en/1.1.x/

# 4)Документация Webix: https://docs.webix.com

# 5)Исходный код приложения: https://github.com/moevm/nosql2h19-rosbag

# ПРИЛОЖЕНИЕ А

# ИНСТРУКЦИЯ ПО СБОРКЕ И ЗАПУСКУ