

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: ИНДЕКС РИФМ

Студенты гр. 6303

Горбунова А.

Жахин А.

Малышенко Ю.

Преподаватель

Заславский М.М.

Санкт-Петербург

2019

ЗАДАНИЕ

Студенты

Горбунова А.

Жахин А.

Малышенко Ю.

Группа 6303

Тема проекта: Разработка приложения для поиска рифм

Исходные данные:

Необходимо реализовать приложение для поиска рифмующихся строк в песнях для СУБД MongoDB.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Выводы»

«Приложение»

«Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студенты гр.6303

Горбунова А.

Жахин А.

Малышенко Ю.

Преподаватель

Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема поиска рифмующихся строк из песен для СУБД MongoDB. Актуальность темы обусловлена тем, что на данный момент имеются сервисы подбора рифмующихся слов, но не рифмующихся строк из песен.

Найти исходный код и всю дополнительную информацию можно по ссылке: [\[2\]](#)

SUMMARY

In the course was planned to develop an application in the team for one of the themes. Topic was chosen searching rhyming lines in different songs for DMS MongoDB. The relevance of the topic is due to the fact that at the moment there are services for the selection of rhyming words, but not rhyming lines from the songs.

Source code and all the additional information can be found at: [\[2\]](#)

Содержание.

Введение.	5
Качественные требования к решению.	5
Сценарии использования.	5
Макет UI.	5
Описание сценариев использования.	8
Поиск и удаление песни.	8
Добавление песни	9
Модель данных.	9
NoSQL модель данных (MongoDB).	9
Общая структура	9
Структура документа.	9
Вычисление примерного объема данных	9
Запросы.	10
SQL модель данных.	11
Разработанное приложение.	13
Краткое описание.	13
Схема экранов приложения.	14
Использованные технологии.	14
Ссылки на Приложение.	14
Выводы.	15
Результаты.	15
Приложение.	16
Документация по сборке и разворачиванию приложения.	16
Снимки экранов приложения.	16
Список использованных источников.	18

Введение.

Цель работы – создать решение для хранения песен и для поиска по заданному слову рифмующихся строк песен из базы.

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

Качественные требования к решению.

Требуется разработать приложение с использованием СУБД MongoDB.

Сценарии использования.

Макет UI.

Макет – [3]

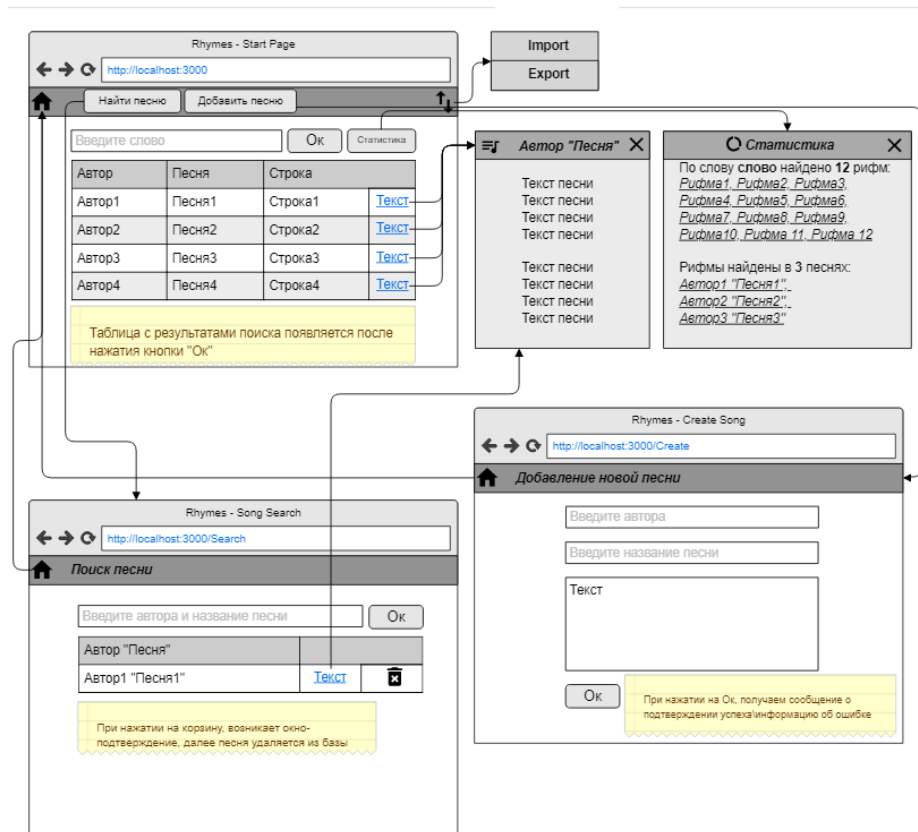


Рисунок 1 – Общий вид.

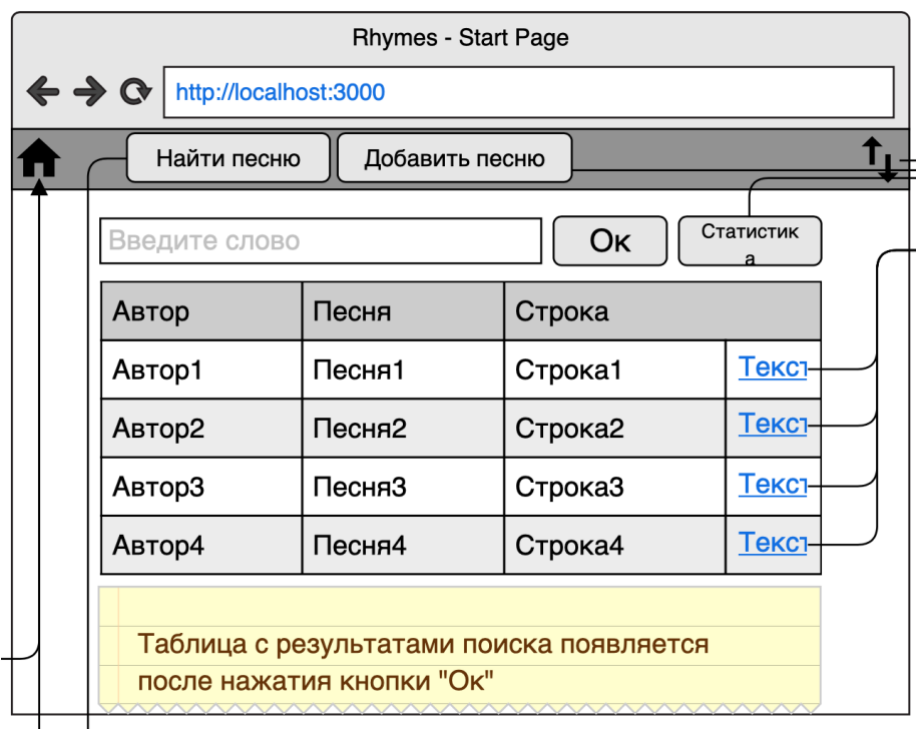


Рисунок 2 – Главная страница

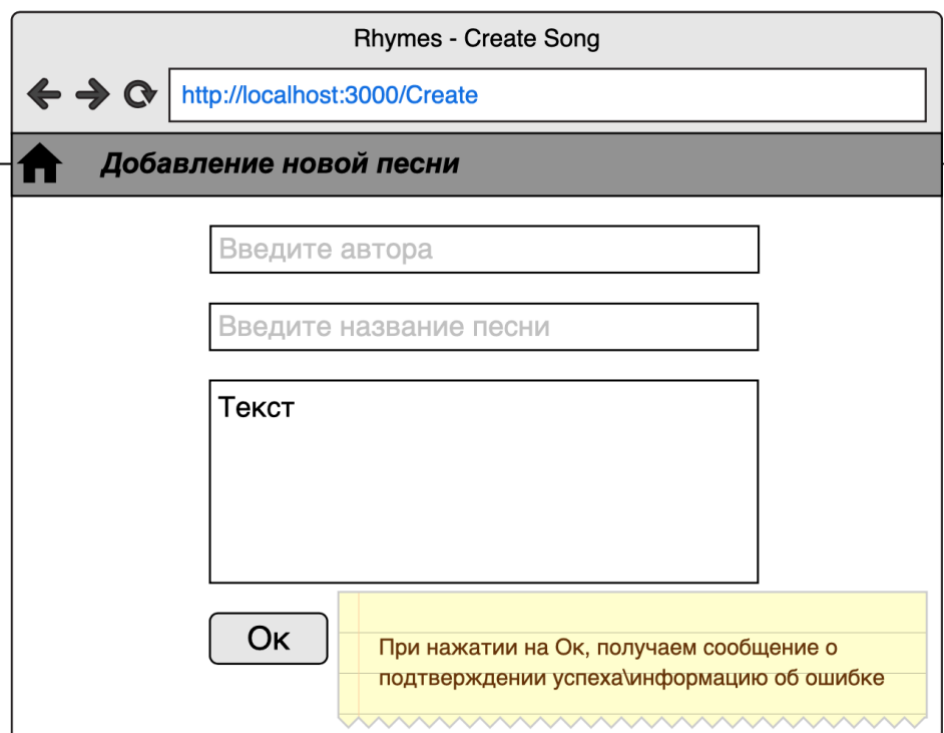




Рисунок 3 – Добавление новой песни

Rhymes - Song Search



← → ↻

 **Поиск песни**

Автор "Песня"	
Автор1 "Песня1"	Текст 



При нажатии на корзину, возникает окно-подтверждение, далее песня удаляется из базы

Рисунок 4 – Поиск песен по базе данных

➔  **Автор "Песня"** 

Текст песни
Текст песни
Текст песни
Текст песни

Текст песни
Текст песни
Текст песни
Текст песни

 **Статистика** 

По слову **слово** найдено **12** рифм:
Рифма1, Рифма2, Рифма3,
Рифма4, Рифма5, Рифма6,
Рифма7, Рифма8, Рифма9,
Рифма10, Рифма 11, Рифма 12

Рифмы найдены в **3** песнях:
Автор1 "Песня1",
Автор2 "Песня2",
Автор3 "Песня3"

Рисунок 5 – Текст песни и статистика

Описание сценариев использования.

Поиск рифм

Пользователь производит поиск строк-рифм по слову. Результатом поиска будут строки из песен, рифмующиеся с данным словом. Можно просмотреть полный текст песни, из которой взята строка, а также можно посмотреть некоторые статистики: общее количество строк, рифмующихся к данному слову, полный список слов-рифм и список песен, в которых встречались рифмы.

Порядок действий:

1. Пользователь заходит на сайт.
2. Вводит слово в input, рифмы к которому он хочет найти.
3. Далее при нажатии на button на сервер отправляется запрос.
4. Пользователю выводится таблица, содержащая автора песни, ее название и строку, последнее слово которой, рифмуется с искомым. При нажатии на link рядом со строкой, появляется окно, в котором можно посмотреть полный текст.
5. Чтобы посмотреть полную статистику, Пользователь нажимает на button «Статистика» на той же странице.

Поиск и удаление песни.

Пользователь ищет конкретные песни в базе данных, может посмотреть их текст или удалить их из базы.

Порядок действий:

1. Пользователь заходит на сайт.
2. Переходит на страницу поиска песни.
3. Вводит информацию о песне в некоторый input.
4. Далее при нажатии на button на сервер отправляется запрос.
5. Пользователь видит найденную песню в таблице, иначе получает сообщение о том, что такой песни нет.
6. При нажатии на link рядом с песней, появляется окно, в котором можно посмотреть полный текст. При нажатии на корзину, возникает предупреждение об удалении, далее песня удаляется из базы.

Добавление песни

Пользователь добавляет песню в базу данных, чтобы по ней также проводился поиск.

Порядок действий:

1. Пользователь заходит на сайт.
2. Переходит на страницу добавления песни.
3. Вводит информацию о песне в некоторые input-ы (автор, название песни, текст песни).
4. Отправляет песню на сервер нажатием на button, где она сохраняется, и пользователь видит подтверждение успеха, либо неудачи.

Модель данных.

NoSQL модель данных (MongoDB).

Общая структура

Данные хранятся в MongoDB. В базе данных всего одна коллекция Songs.

Структура документа

Для определения размера документа условимся, что у нас используются только ASCII-символы. Документ содержит:

- Идентификатор
 - Название песни
 - Исполнителя
 - Текст строки песни
 - Последнее слово строки
- Пример JSON:

```
{ "author": "....", "title": "...", "string": "...", "lastword": "...",  
  "_id": 123 }
```

Вычисление примерного объема данных

_id - Поле, которое автоматически генерируется MongoDB, его размер 12-байт, судя по документации. Будем считать название песни размером в 15 символов. Имя автора также будем считать как 15 символов. Кол-во

символов в строке лежит в среднем где-то между 30-50, зависит от многих условий, примем его равным 40. Кол-во символов в слове лежит в среднем где-то между 5-10, возьмем худший случай, равный 10 символам.

- $_id - V_{id} = 12b$
- $title - string \Rightarrow V_t = 2b * N_t$, где $N_t = 15$ в среднем $\Rightarrow V_t = 30b$
- $author - string \Rightarrow V_a = 2b * N_a$, где $N_a = 15$ в среднем $\Rightarrow V_a = 30b$
- $string - string \Rightarrow V_s = 2b * N_s$, где $N_s = 40$ в среднем $\Rightarrow V_s = 80b$
- $lastword - string \Rightarrow V_w = 2b * N_w$, где $N_w = 10$ в среднем $\Rightarrow V_p = 20b$

Приняв во внимание то, что написано выше, мы получим, что средний размер документа: $sizeof(id) + sizeof(title) + sizeof(artist) + sizeof(string) + sizeof(lastword) = 12 + 30 + 30 + 80 + 20 = 172$, следовательно, размер одного документа будет равен 172-байта. $V_{str}=172b$

В одном документе лежит только одна строка песни, рассчитаем размер всей коллекции (всех песен $N \sim 1000$).

Возьмем среднее количество строк в каждой песне равным 40. Тогда общий размер коллекции будет равен: $V(N) = N_{str} * V_{str} * N$, где N_{str} это количество строк в одной песне $V(N) = 40 * 172b * 1000 = 6880000b = 6,88Mb$

$$V(N) = 6880 * N = 6880000b = 6,88 Mb$$

Избыточность:

Вычислим память для непострочного хранения N песен, содержащих по 40 строк. Длина строки ~ 40 символов:

$$V(N) = N_{str} * V_s * N$$

$$V(N) = 3200 * N = 3200000 b = 3,2 Mb$$

Запросы

- Добавление Пример запроса на создание одного документа, хранящего в себе строку:

```
Collection.insert ({title: "Yesterday", author: "The Beatles", string: "Love was such an easy game to play", lastword: "play"})
```

Таким образом формула количества запросов для одной песни: $Q_N = N_{str}$, где N_{str} - количество строк в песне (~ 40).

Количество запросов для N песен, где $N \sim 1000$:

$$Q_N(N) = N * N_{str} = 40 * 1000 = 40000 \text{ запросов}$$

$QN(N) = 40 * N = 40000$ запросов

- Поиск Пример запроса на поиск по коллекции:

```
collection.find({title: "Yesterday", author: "The Beatles"})
```

- Удаление Пример запроса на удаление всей песни:

```
collection.remove({title: "Yesterday", author: "The Beatles"})
```

SQL модель данных.

Общая структура

Нам понадобится несколько сущностей: Author, Song, также надо учесть, что исполнители могут писать одну песню вместе, т.е. между этими сущностями будет отношение many-to-many, т.е. составим таблицы:

```
CREATE TABLE Author ( id SERIAL, name VARCHAR(15) NOT NULL, CONSTRAINT PK_AUTHOR PRIMARY KEY (id) );
```

```
CREATE TABLE Song ( id SERIAL, name VARCHAR(15) NOT NULL, string VARCHAR(40) NOT NULL, lastWord VARCHAR(10) NOT NULL, CONSTRAINT PK_SONG PRIMARY_KEY (id) )
```

```
CREATE TABLE SongToAuthor ( song_id INTEGER NOT NULL REFERENCES Song, author_id INTEGER NOT NULL REFERENCES Author, )
```



Рисунок 6 – Структура SQL модели

Вычисление примерного объема данных

1) Таблица "Author"

- author_id - INT 4b

- name - VARCHAR(N), по байту на CHAR N ~ 15
- Итог: $V_a = 19b$ на строку в таблице "Author"

2) Таблица "Song"

- song_id - INT 4b
 - title - VARCHAR(N), по байту на CHAR N ~ 15
 - string - VARCHAR(N), по байту на CHAR N ~ 40
 - lastword - VARCHAR(N), по байту на CHAR N ~ 10
- Итог: $V_s = 69b$ на строку в таблице "Song"

3) Таблица "SongToAuthor"

- author_id - INT 4b
- song_id - INT 4b

Итог: $V_{as} = 8b$ на строку в таблице "SongToAuthor"

Для сравнения допустим, что у нас 1000 песен и 20 исполнителей, т.е. у исполнителя примерно 50 песен, это значит, что $N = 1000$, а $k = 20$. Таким образом при количестве строк N_{str} в песне примерно равным 40 получаем:

$$V(N) = N * N_{str} * V_s + k * V_a + N * V_{as} = 2768380b = 2,77Mb$$

$$V(N) = 2777 * N + 380 = 2,77Mb$$

Здесь SQL выигрывает.

Запросы

Добавление.

Пример запроса на добавление записи в таблицу "Author":

```
INSERT INTO Author (name) values('The Beatles');
```

Пример запроса на добавление записи в таблицу "Song":

```
INSERT INTO Song (title, string, lastword) values('Yesterday', 'Love was such an easy game to play ', play);
```

Пример запроса на добавление записи в таблицу "SongToAuthor":

```
INSERT INTO SongToAuthor (author_id, song_id) values('The Beatles', Yesterday);
```

Таким образом формула количества запросов для одной песни:
 $Q_N = 1 + 1 + N_{str}$,

где Nstr - количество строк в песне (~40).

Количество запросов для N песен, где $N \sim 1000$:

$QN(N) = N * QN = 42 * N = 42000$ запросов

Поиск

```
SELECT song_id, name, text, name FROM ( SELECT * FROM SongTOArtist  
JOIN Song ON Song.id == song_id JOIN Artist ON Artist.id == artist_id  
);
```

Тут видно, что в MongoDB мы делаем это за $O(n)$, а в SQL будет $O(n + (n / k) \log(n / k)) \sim O(n \log(n))$

Удаление

```
DELETE TABLE Song WHERE id == $id;
```

В двух случаях оценка сложности $O(n)$, т.к. нету JOIN.

Сравнение SQL и NoSQL

- В SQL реализации модели данных пришлось бы создавать дополнительные таблицы для связей, что увеличивает суммарное количество создаваемых таблиц.
- Данные в SQL модели будут занимать меньший объем (для 1000 песен 2.77 Mb), по сравнению с данными в NoSQL модели (для 1000 песен 6.88 Mb).
- Количество запросов для NoSQL модели данных гораздо меньше чем для SQL

NoSQL реализация будет выигрывать в структуре и скорости сложных запросов.

Разработанное приложение.

Краткое описание.

Веб-приложение, хранящее в себе песни с возможностью поиска рифмующихся строчек по заданному пользователем слову. А также с возможностью добавлять и удалять песни и просматривать некоторую статистику.

Схема экранов приложения.



Рисунок 7 – Схема экранов приложения

Использованные технологии.

- MongoDB
- JavaScript
- NodeJS
- HTML

Ссылки на Приложение.

Ссылка на github: [\[2\]](#)

Выводы:

В ходе работы было разработано приложение, позволяющее пользователю осуществлять поиск рифмующихся к заданному слову строчек из песен, хранящихся в базе данных приложения. Так же, была реализована возможность добавлять и удалять песни, а также просматривать текст песни целиком. В приложение была добавлена статистика по найденным рифмам. В учебных целях были добавлены `import` и `export` датасетов.

Приложение.

Документация по сборке и разворачиванию приложения.

- 1) Скачать проект из репозитория.
- 2) Открыть с помощью Visual Studio Code
- 3) В терминале прописать следующие команды:

```
$ npm install
$ npm install cookie-parser
$ npm run start
```

- 4) Перейти по <http://localhost:3001>.

Снимки экранов приложения.

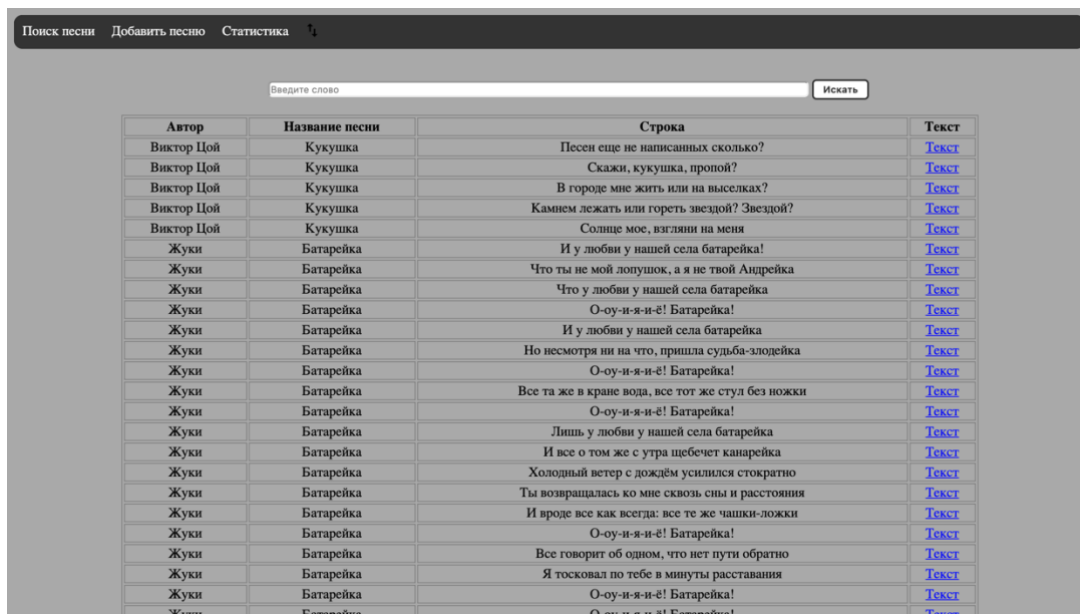


Рисунок 22 – Главная страница

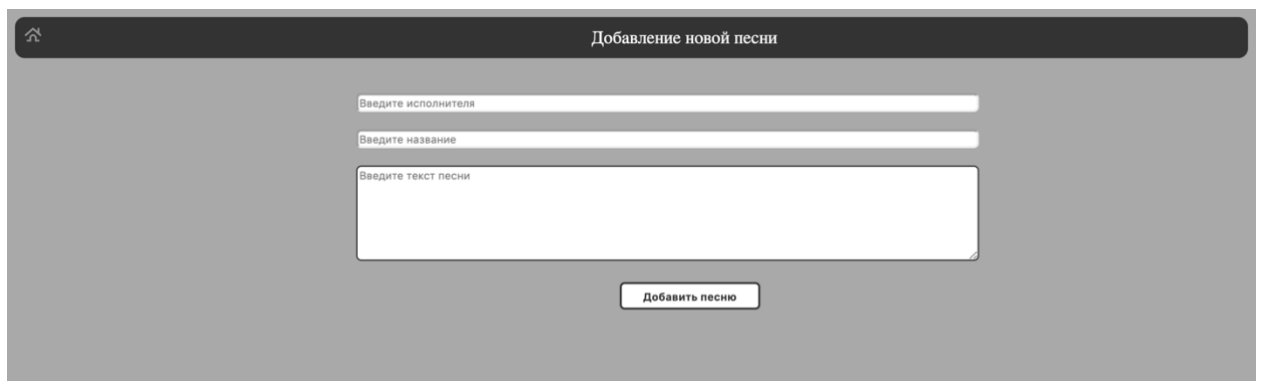


Рисунок 23 – Добавление новой песни

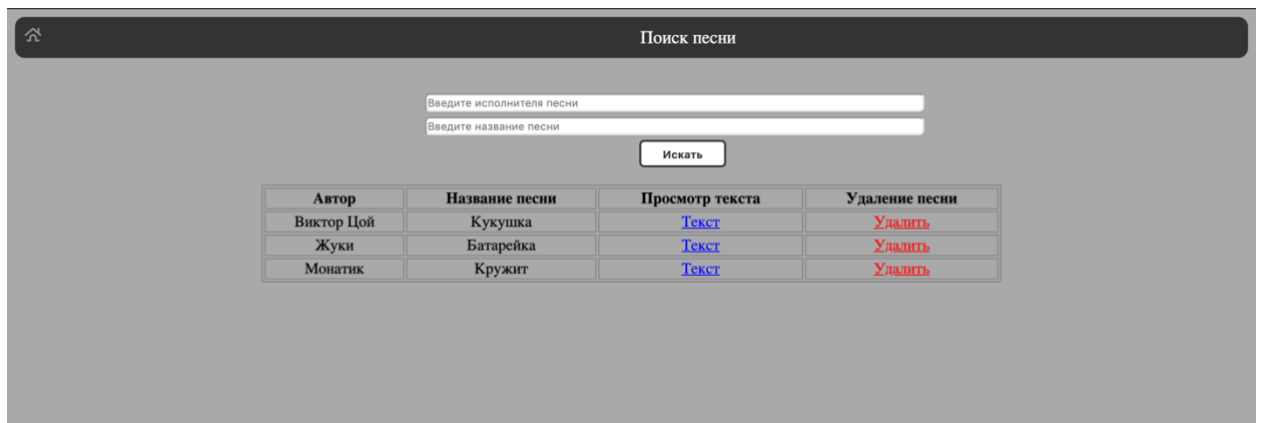


Рисунок 24 – Поиск песни по базе



Рисунок 25 – Статистика по количеству слов-рифм во всех песнях

Статистика по количеству слов-рифм в каждой песне.

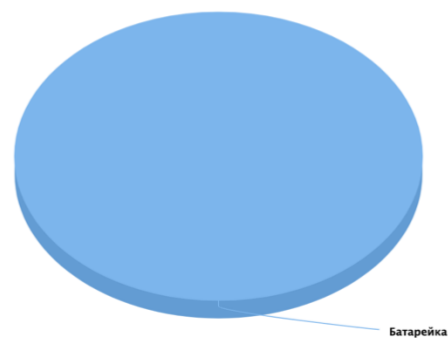


Рисунок 26 – Статистика по количеству слов-рифм в каждой песне

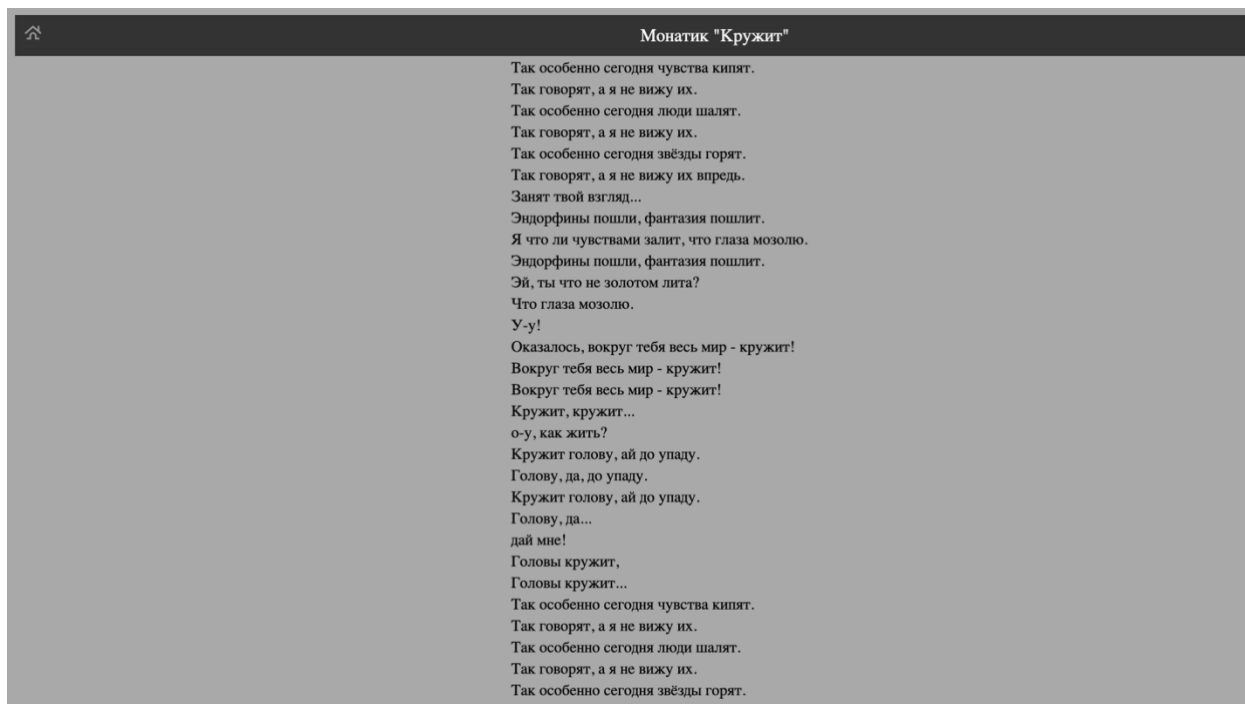


Рисунок 27 – Страница с текстом одной из песен

Список использованных источников.

1. Документация MongoDB: <https://docs.mongodb.com>
2. Репозиторий проекта: <https://github.com/moevm/nosql1h19-rhymes-index>
3. Макет UI <https://app.moqups.com/d6yZJT8q0Z/view/page/aa9df7b72>