**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ ФКТИ**

**Индивидуальное домашнее задание**

**по дисциплине «Введение в нереляционные базы данных»**

**Тема:** **«Граф синонимов / антонимов/ словоформ»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 7303 |  | Шестопалов Р.П. |
|  |  | Никитенко Д.А. |
|  |  | Романенко М.В. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2020

**Задание**

Студенты

Шестопалов Р.П.

Никитенко Д.А.

Романенко М.В.

Группа 7303

Тема проекта: Граф синонимов / антонимов/ словоформ.

Исходные данные:

Необходимо реализовать приложение для поиска, добавления и удаления синонимов, антонимов и словоформ на основе СУБД Neo4j.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарий использования»

«Модель данных»

«Разработка приложения»

«Вывод»

«Приложение»

Предполагаемый объем пояснительной записки: не менее 10 страниц.

Дата выдачи задания:

Дата сдачи ИДЗ:

Дата защиты ИДЗ:

|  |  |  |
| --- | --- | --- |
| Студент гр. 7303 |  | Шестопалов Р.П. |
|  |  | Никитенко Д.А. |
|  |  | Романенко М.В. |
| Преподаватель |  | Заславский М.М. |

**Аннотация**

В данной курсовой работе представлены этапы разработки веб-приложения, которое строит граф слов и их взаимных отношений, предоставляет интерфейсы для поиска, построения путей, сопоставления с текстами на естественном языке. Исходный код находится здесь: <https://github.com/moevm/nosql2h20-synonims>

Содержание

[1. Введение 6](#_Toc60073676)

[2. Качественные требования к решению 6](#_Toc60073677)

[3. Сценарии использования 6](#_Toc60073678)

[4. Модель данных 9](#_Toc60073679)

[5. Разработанное приложение 13](#_Toc60073680)

[6. Вывод 15](#_Toc60073692)

[7. Приложение 15](#_Toc60073693)

[8. Используемая литература 15](#_Toc60073698)

### **Введение**

Цель работы – создать приложение для поиска синонимов/антонимов/словоформ

Было решено разработать веб-приложение, которое позволит хранить слова и их связи друг с другом и которое будет позволять редактировать этот словарь

1. **Качественные требования к решению**

Требуется разработать приложение с использованием СУБД Neo4J

1. **Сценарии использования**

**Основной сценарий:**

1. Пользователь заходит на сайт для поиска необходимых ему синонимов, антонимов или словоформ
2. Пользователь нажимает на кнопку "Пуск" и переходит на рабочую страницу
3. Пользователь вводит необходимое слово
4. Пользователь в checkbox'ах выбирает необходимую ему информацию (синонимы, антонимы и/или словоформы)
5. Пользователь нажимает кнопку "Поиск"
6. Пользователю выводится необходимая информация.
7. Переход на шаг 3.

**Альтернативный сценарий:**

* Если искомое слово не найдено, то пользователю выводится сообщение "Данное слово не найдено"
* Пользователь нажимает кнопку "Ок"
* Переход на шаг 3 основного сцена

**Сценарий добавления новых слов:**

1. Пользователь заходит на сайт
2. Пользователь нажимает на кнопку "Импорт/Экспорт"
3. Пользователь нажимает на кнопку "Импорт"
4. Пользователь выбирает файл для массового импорта в словарь

**Сценарий загрузки словаря на локальную машину:**

1. Пользователь заходит на сайт
2. Пользователь нажимает на кнопку "Импорт/Экспорт"
3. Пользователь нажимает на кнопку "Экспорт"
4. Пользователь выбирает куда загрузить словарь

**Сценарий получения статистики:**

1. Пользователь заходит на сайт
2. Пользователь нажимает на кнопку "Статистика"
3. Пользователь выбирает тип статистики

**Сценарий добавления новых пар синонимов, антонимов, словоформ:**

1. Пользователь заходит на сайт
2. Пользователь нажимает кнопку "Добавление/удаление слов"
3. Пользователь нажимает кнопку "Добавить"
4. Пользователь вводит в формы новые пары и нажимает кнопку "Добавить"

**Сценарий удаления слова или связи синонимов, антонимов, словоформ:**

1. Пользователь заходит на сайт
2. Пользователь нажимает кнопку "Добавление/удаление слов"
3. Пользователь нажимает кнопку "Удалить"
4. Пользователь выбирает из списка слово или связь, которую он хочет удалить

**Сценарий редактирования слова или связи синонимов, антонимов, словоформ:**

1. Пользователь заходит на сайт
2. Пользователь нажимает кнопку "Добавление/удаление слов"
3. Пользователь нажимает кнопку "Редактировать"
4. Пользователь нажимает кнопку "Добавление/удаление слов"

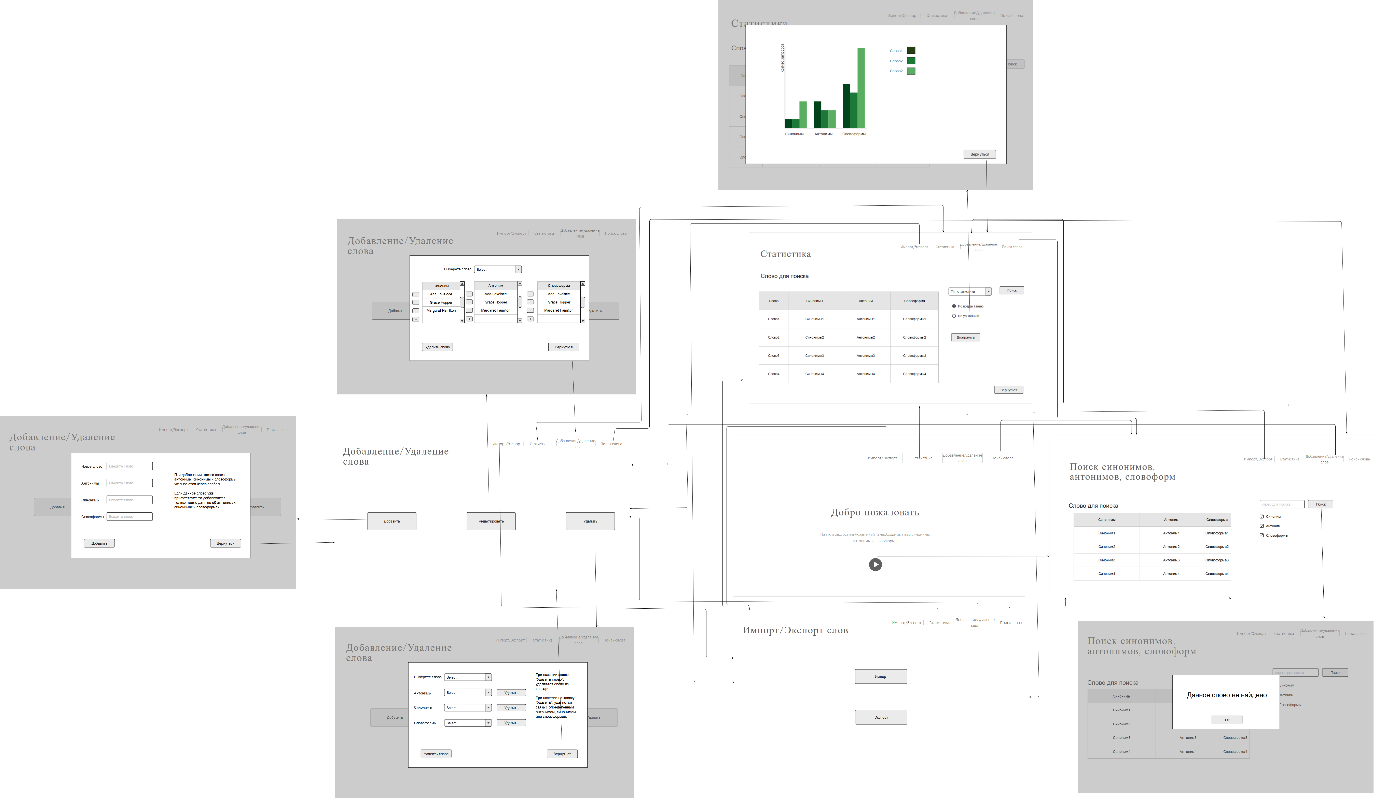
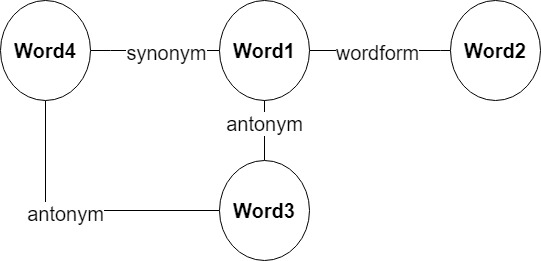


Рисунок 1 - Прототип приложения

1. **Модель данных**

**NoSQL**

**Графическое представление**



**Описание сущностей и типов данных**

**Сущность Word:**

* id - тип Int: 4 Byte.
* Слово/word - строковый тип String. Средняя длина b символов.

**Существует 3 связи между сущностями:**

* антоним/antonym
* синоним/synonym
* словоформа/wordform Связи не содержат дополнительных атрибутов.

**Оценка удельного объема информации, хранимой в модели**

Предположим, что у каждого слова 2 синонима, 1 антоним и 4 словоформ. Тогда на каждое слово приходится 3,5 связи. Размер символа — 2 байта. Связь хранит идентификаторы вершин, которые связывает, и название связи(10 \* 2B). Будем считать, что всего N слов, а средняя длина слова b - 6 символов. Тогда фактический размер базы данных:

(4+6\*2)N + (4\*2+20)3.5N = 144N

При 1000 слов объем информации будет равен: 144000 байт

**Избыточность модели**

Модель избыточна, так как мы храним одинаковые названия связей и идентификатор для слов. "Чистые" данные будут занимать: 40N

Вычислим отношение фактического и «чистого» объемов данных:

144/40 = 3.6

**Направление роста модели**

Линейный рост, при добавлении:

Для слов - 16B

Для связей - 28B

**Запросы к модели, с помощью которых реализуются сценарии использования**

* Добавление нового слова

CREATE (n:Word {id: 1, word: "hello"})

* Создание связи

MATCH (a:Word),(b:Word)

WHERE a.id = '...' AND b.id = '...'

CREATE (a)-[r:synonym]-(b)

RETURN type(r)

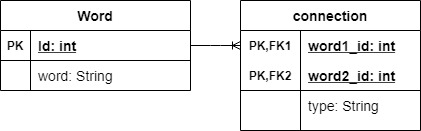
* Поиск всех антонимов для слова

MATCH (:Word {word: "hello"})-[:antonym]-(antonym:Word)

RETURN antonym.word

**SQL модель**

**Графическое представление**



**Описание сущностей и типов данных**

**Таблица word:**

* id - тип Int: 4 Byte.
* Слово/word - строковый тип String. Средняя длина b символов.

**Таблица connection:**

* word1\_id - тип Int: 4 Byte.
* word2\_id - тип Int: 4 Byte.
* тип/type - строковый тип String. Средняя длина k символов.

**Оценка удельного объема информации, хранимой в модели**

В отличии от NoSQL модели в реляционной модели будет в 2 раза больше связей, так как мы будем хранит связь для каждого из 2 слов.  
Предположим, что у каждого слова 2 синонима, 1 антоним и 4 словоформ. Тогда на каждое слово приходится 7 связей. Размер символа — 2 байта. Связь хранит идентификаторы вершин, которые связывает, и название связи(10 \* 2B). Будем считать, что всего N слов, а средняя длина слова b - 6 символов. Тогда фактический размер базы данных:

(4+6\*2)N + (4\*2+20)7N = 212N

При 1000 слов объем информации будет равен: 212000 байт

**Избыточность модели**

Модель избыточна, так в таблице connection можно вынести тип в отдельную структуру. "Чистые" данные будут занимать: 100N

Вычислим отношение фактического и «чистого» объемов данных:

212/100 = 2.12

**Направление роста модели**

Линейный рост, при добавлении:

Для слов - 16B

Для связей - 56B

**Запросы к модели, с помощью которых реализуются сценарии использования**

* Добавление нового слова

INSERT INTO Word VALUES(...)

* Создание связи

INSERT INTO connection VALUES(...)

* Поиск всех антонимов для слова

SELECT "Word" FROM connection WHERE type = "antonim"

**Сравнение SQL и NoSQL**

* В SQL реализации модели данных пришлось бы создавать дополнительные таблицы для связей, что увеличивает суммарное количество создаваемых таблиц.
* В SQL версии данные занимают больше места.
* Количество запросов, необходимых для выполнения юзкейсов в SQL модели больше.

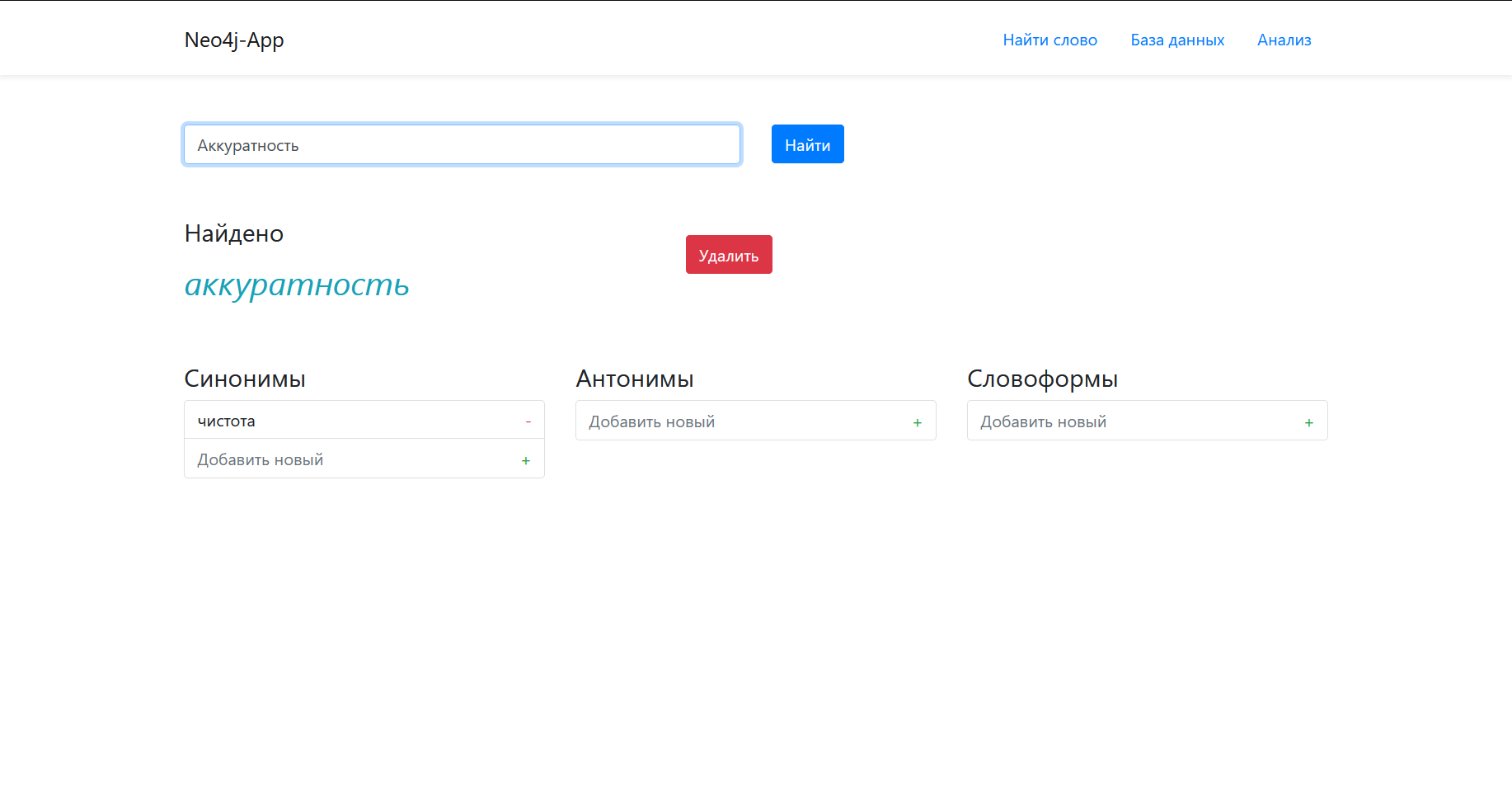
1. **Разработанное приложение**

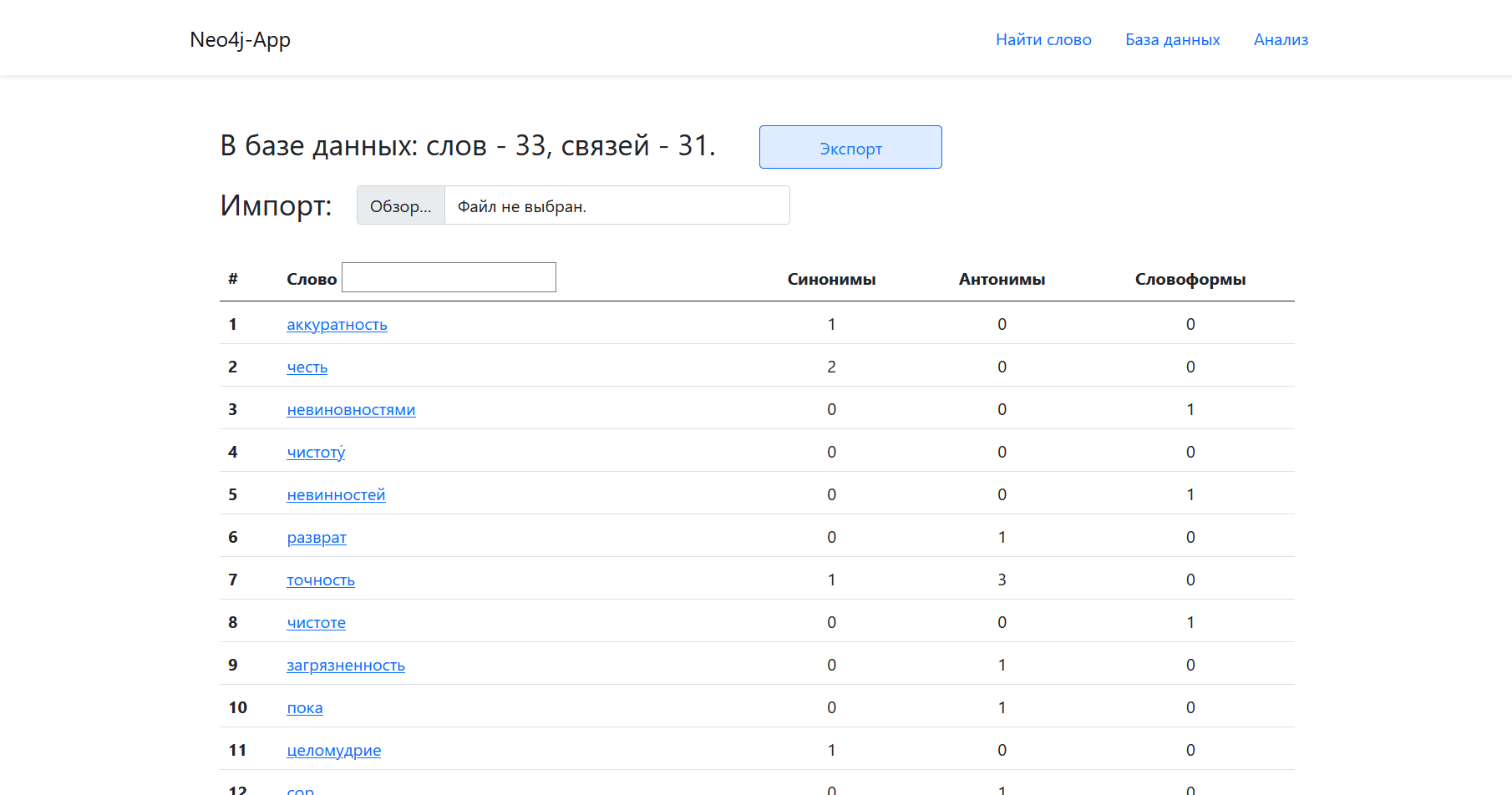
**Краткое описание**

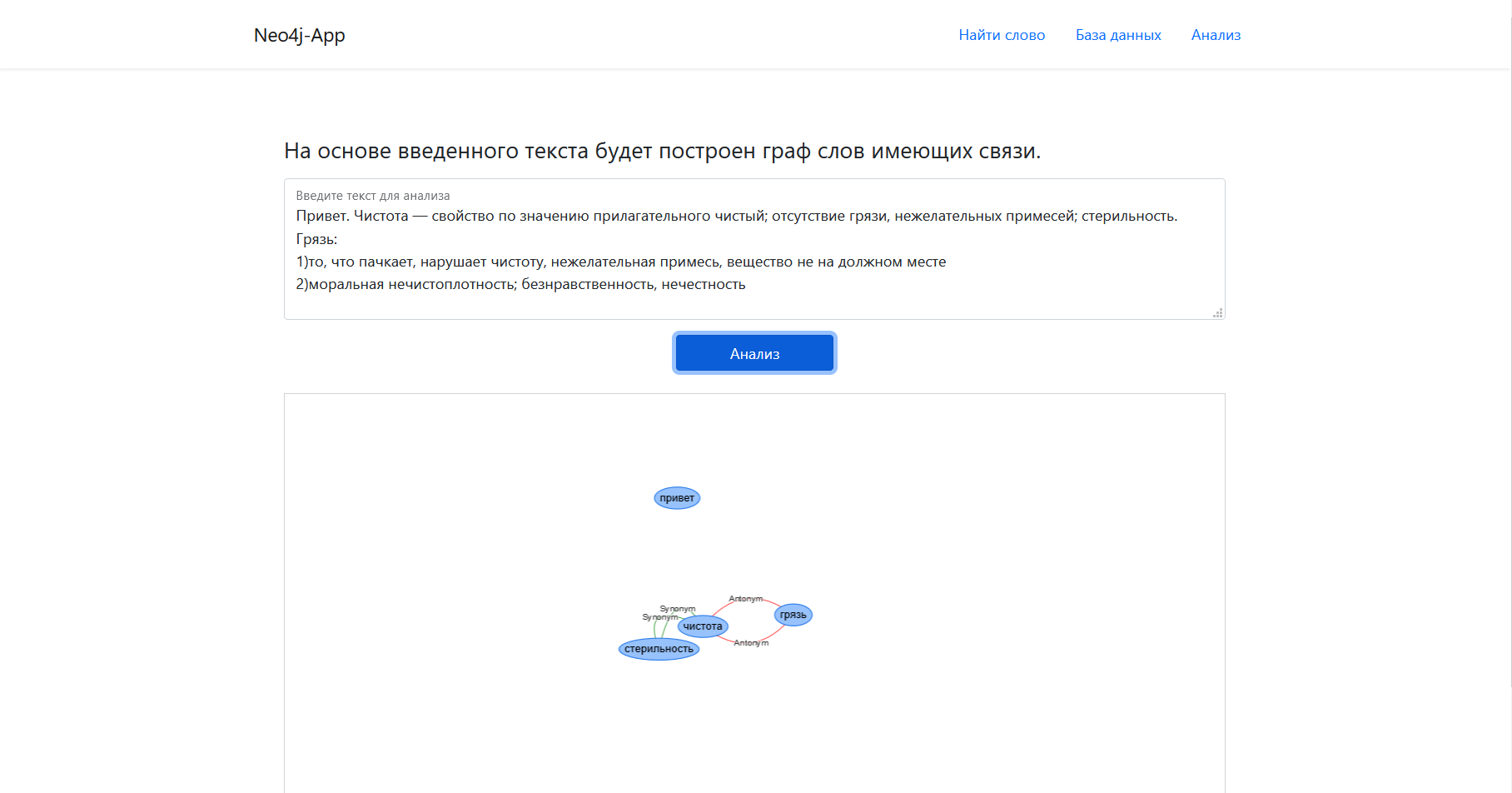
Back-end представляет из себя node.js приложение.

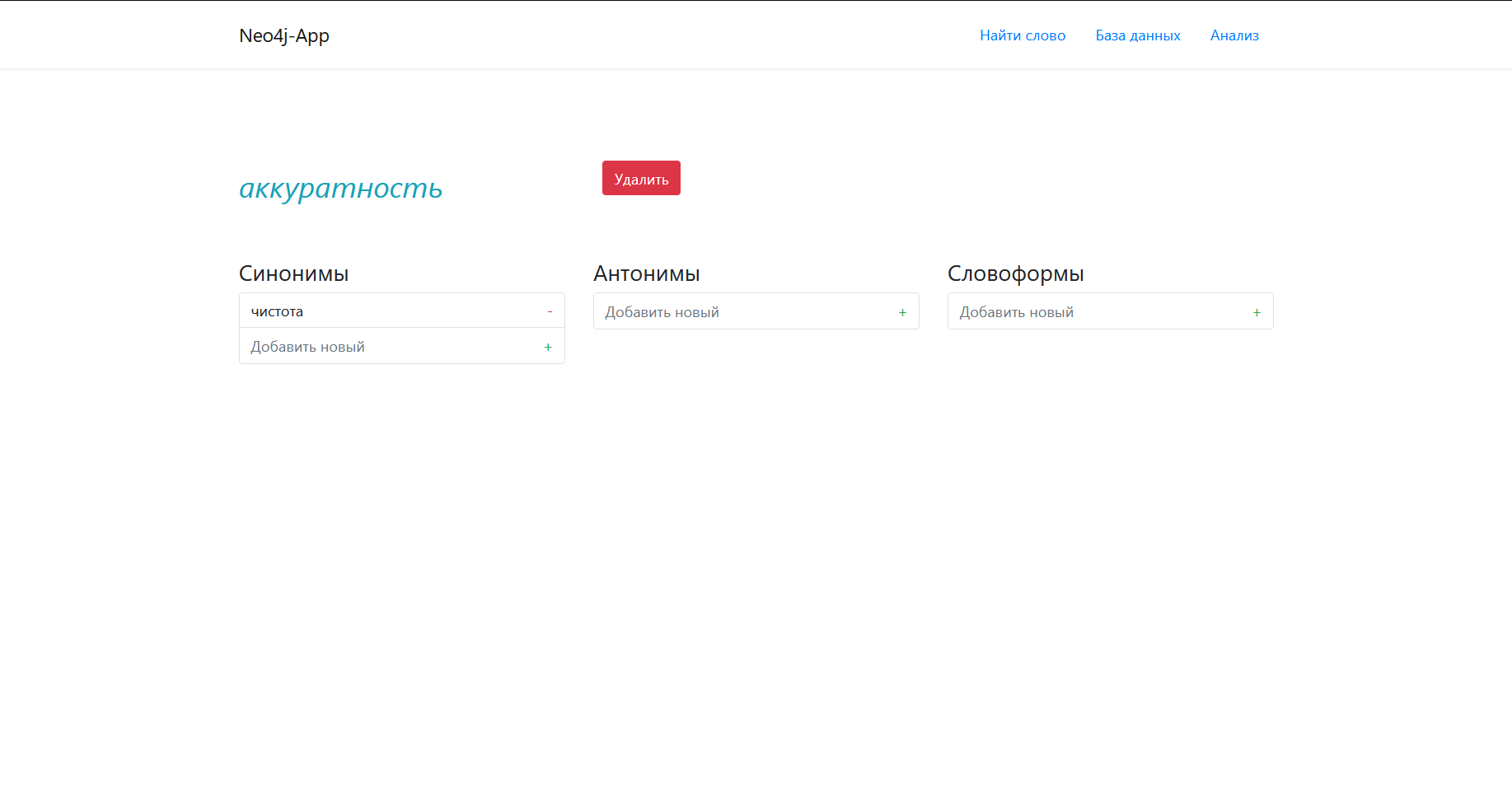
Front-end – веб-приложение, которое использует API back-end’а и отображает данные в удобном для пользователя виде.

**Схема экранов приложения**









**Использованные технологии**

БД: Neo4J

Back-End: node.js

Front-End: HTML, CSS, JavaScript

**Ссылка на приложение**

* 1. <https://github.com/moevm/nosql2h20-synonims>

1. **Вывод**

В ходе работы было разработано приложение для добавления, удаления слов и редактирования словаря синонимов/антонимов/словоформ.

1. **Приложение**
2. Скачать проект из репозитория
3. В папке App запустить терминал
4. В терминале ввести команду «docker-compose up»
5. Открыть приложение в браузере по адресу localhost:3000
6. **Используемая литература**

1. Документация Neo4J: https://neo4j.com/docs/