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: Разработка приложения для анализа медицинских данных

Студенты гр. 6382 _____ Вайгачев А.О.

_____ Мартыненко П.П.

_____ Шарыпина Д.В.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2019

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студенты гр. 6382 Вайгачев А.О., Мартыненко П.П., Шарыпина Д.В.

Группа 6382

Тема проекта: Разработка приложения для анализа медицинских данных

Исходные данные:
Реализовать приложение, использующее СУБД Neo4j

Содержание пояснительной записки:
«Содержание», «Введение», «Качественные требования к решению», «Сценарии 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:

Студенты гр. 6382 _____ Вайгачев А.О.

_____ Мартыненко П.П.

_____ Шарыпина Д.В.

Преподаватель _____ Заславский М.М.

АННОТАЦИЯ

В рамках данного курса было необходимо реализовать приложение на одну из поставленных тем. Была выбрана тема «Анализ медицинских данных». В результате работы было разработано приложение, которое хранит данные о симптомах и болезнях, предоставляет возможность по заданным симптомам получить вероятности возможных болезней, а также посмотреть статистику по данным.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью работы является реализация приложения для хранения и анализа медицинских данных.

Было решено с использованием СУБД Neo4j разработать приложение, которое позволит хранить информацию о симптомах и о соответствующих им болезнях, а также получать статистику по хранящимся данным.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Были выдвинуты следующие качественные требования к разрабатываемому приложению:

- Приложение осуществляет хранение и обработку данных с помощью СУБД Neo4j.
- Приложение предоставляет возможность поиска симптомов.
- Приложение реализует вычисление вероятности болезней по заданным симптомам.
- В приложении имеется возможность импорта и экспорта данных.
- Приложение предоставляет возможность подсчета следующей статистики по данным:
    - Самые размытые симптомы;
    - Самые сложные для диагностики болезни;
    - Среднее количество симптомов для болезней.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет пользовательского интерфейса

На рисунке 1 представлен интерфейс для поиска симптомов с выводом списка соответствующих болезней с вероятностями их диагностики.

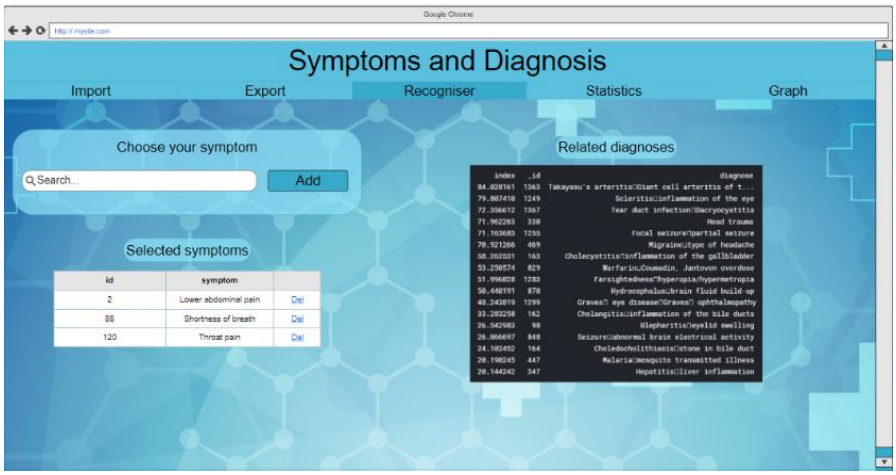

Рисунок 1 – Интерфейс для поиска симптомов

На рисунке 2 представлен интерфейс окна для импорта данных о симптомах и болезнях, а также о связях между ними.

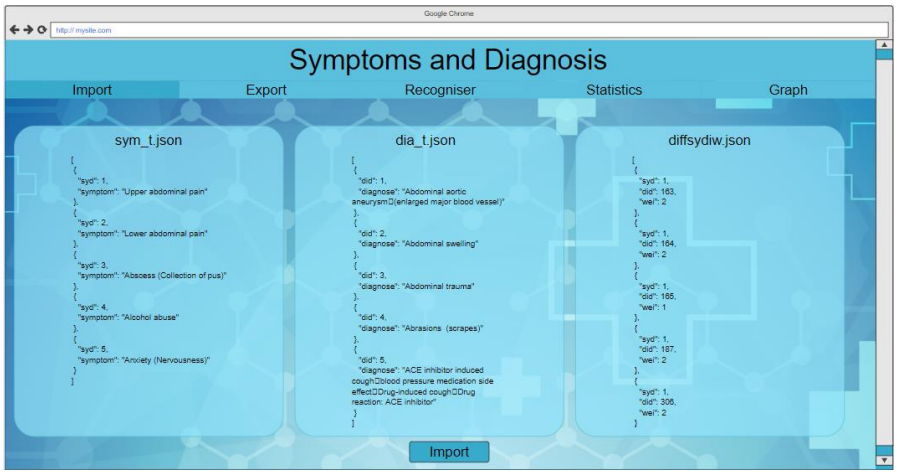

Рисунок 2 – Интерфейс для импорта данных

На рисунке 3 представлен интерфейс окна для экспорта данных о симптомах и болезнях.

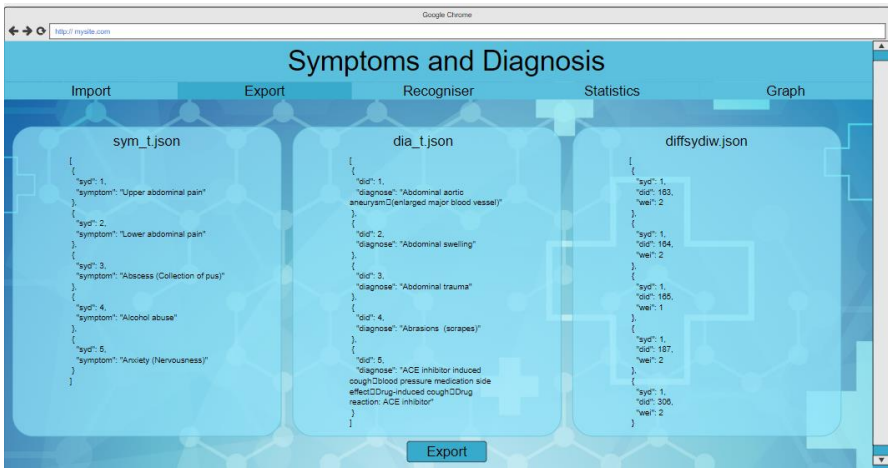

Рисунок 3 – Интерфейс для экспорта данных

На рисунке 4 представлен интерфейс для выбора информации для подсчета статистики.

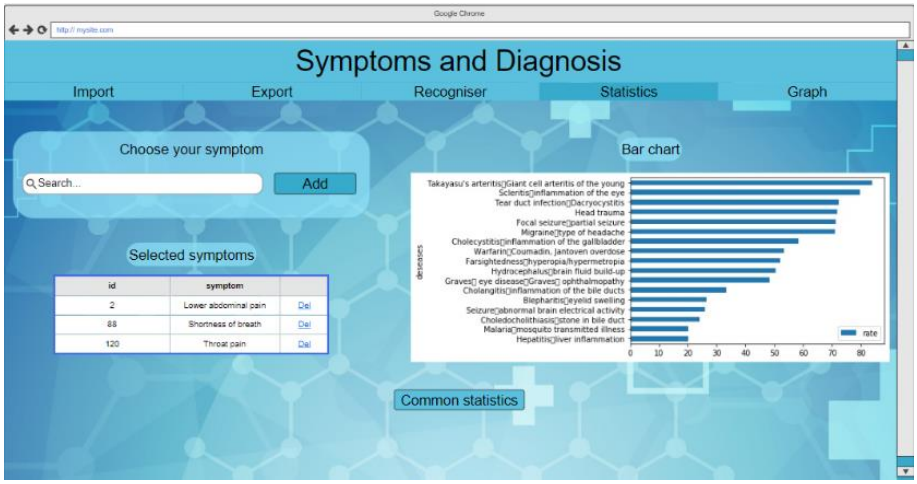

Рисунок 4 – Интерфейс для подсчета статистки

На рисунке 5 представлен интерфейс окна для отображения вычисленной статистики по данным.

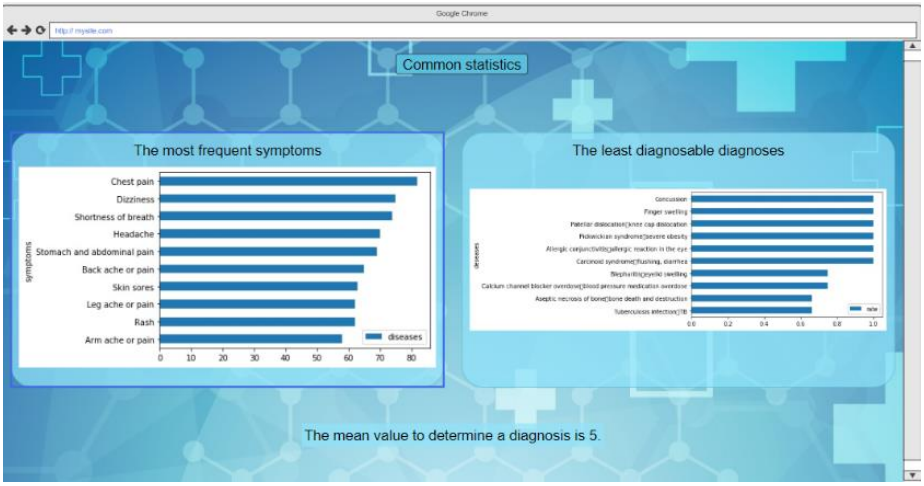

Рисунок 5 – Отображение статистики по данным

2.2. Сценарии использования

1) Сценарий использования – «Узнать болезнь»

   Действующее лицо: Пользователь, приложение

   Основной сценарий:

   a. Пользователь открывает главную панель
   b. Пользователь вводит симптомы
   c. Приложение анализирует выборы пользователя
   d. Приложение предоставляет на экране возможное заболевание и их вероятности

   Альтернативный сценарий:

   - Пользователь не смог открыть главную панель

2) Сценарий использования – «Узнать болезнь: Takayasy’s arteris»

   Действующее лицо: Пользователь, приложение

   Возможный сценарий:

   a. Пользователь открывает главную панель
   b. Пользователь вводит свои симпотмы
      i. Lower abdominal pain
      ii. Shortness of breath
      iii. Throat pain
   c. Приложение выводит справа вероятные болезни в виде таблицы, на первом месте которой стоит Takayasy’s arteris

   Альтернативный сценарий:

   - Пользователь ошибся в симптомах из-за чего получил неверный диагноз
   - Загруженная таблица не корректная

3) Сценарий использования – «Получить статистику базы данных»

   Действующее лицо: Пользователь, приложение

   Возможный сценарий:

   a. Пользователь загружает базу данных болезней (см п.3)
   b. Пользователь нажимает на кнопку statistic

c. Приложение выводит на дисплей информацию о самом размытом симптоме и самом размытом диагнозе

4) Сценарий использования – «Загрузить базу данных»

Действующее лицо: Пользователь, приложение

Возможный сценарий:

a. Пользователь нажимает на кнопку import
b. Пользователь находит в файловой системе желаемую БД
c. Пользователь загружает БД

Альтернативный сценарий:

- БД не корректного формата или не поддерживается программой

5) Сценарий использования – «Экспортировать базу данных»

Действующее лицо: Пользователь, приложение

Возможный сценарий:

a. Пользователь нажимает на кнопку export
b. Пользователь находит в файловой системе путь, куда сохранить БД
c. Пользователь получает БД в формате json

2.3. Вывод

В результате можно сделать вывод о том, что в разрабатываемом приложении будут преобладать операции чтения, поскольку требуется осуществлять только поиск различной информации по базе данных.

3. МОДЕЛЬ ДАННЫХ

3.1 Нереляционная модель данных

3.1.1. На рисунке 6 приведено графическое представление нереляционной модели данных.

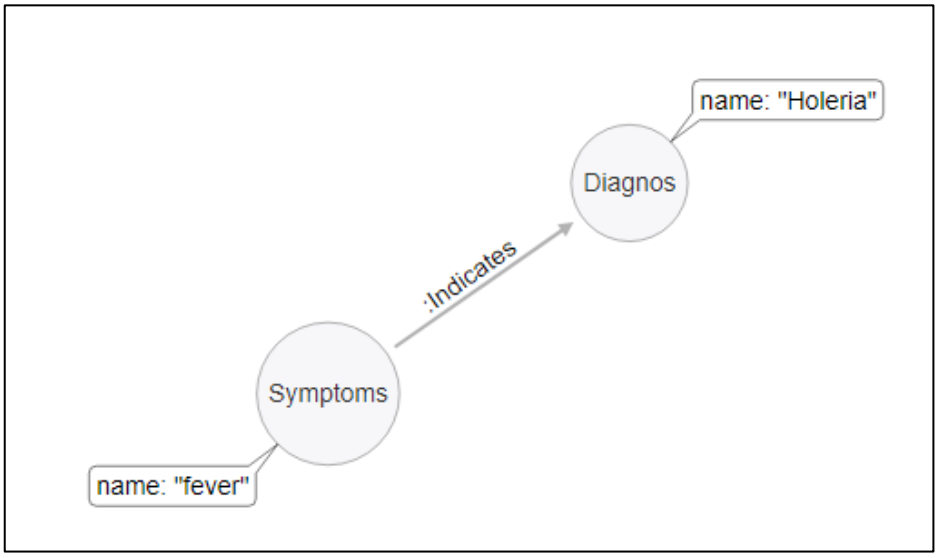

Рисунок 6 – Графическое представление нереляционной модели данных

3.1.2. Описание назначений коллекций, типов данных и сущностей.

В нереляционной модели данных имеется две сущности: Symptom, соответствующая симптомам, а также Diagnosis, соответствующая болезням.

Между сущностью Symptom и сущностью Diagnosis есть односторонняя связь “Indicates”.

Сущности представляют собой вершины в графе со свойством «Name», а связь между сущностями представляет собой однонаправленное ребро графа, также со свойством «Name».

В Neo4j вершины занимают 15 байт, а отношения между ними 34 байта. При этом строковые свойства вершин и отношений занимают 50 байт.

3.1.3. Оценка удельного объема информации, хранимой в модели.

Пусть N – число вершин, R – число связей между вершинами. Т.к. в среднем каждый симптом указывает на 5 болезней, то примем $R = 5N$.

Тогда общий объем $V = (15N + 345N) \cdot 50 = 9250N$

3.1.4. Запросы к модели, с помощью которых реализуются сценарии использования.

а) Узнать болезнь

```
MATCH (s:Symptoms{id: sym_id})-[:INDICATES]->(d:Diagnosis) RETURN
d.name
```

б) Получить статистику

```
MATCH (d:Diagnosis) RETURN d.id AS name, selected_num/size((d)<-
[:INDICATES]-()
```

в) Наиболее часто встречающиеся симптомы

```
MATCH (s:Symptom) RETURN s.id AS name, size((s)-[:INDICATES]->())
LIMIT 10
```

г) Среднее количество симптомов

```
MATCH (d:Diagnosis) RETURN d.id AS name, avg(size((d)<-
[:INDICATES]-()))
```

3.2. Аналог модели для SQL СУБД

3.2.1. Графическое представление.

На рисунке 7 изображено графическое представление SQL модели данных.

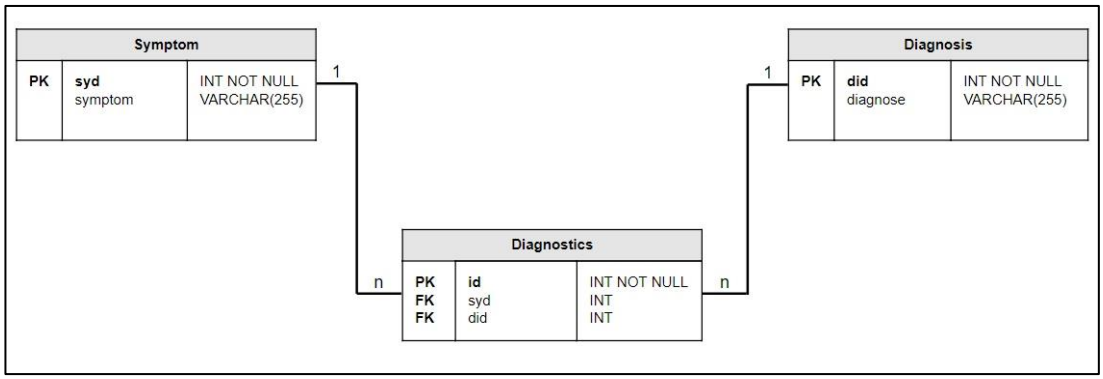

Рисунок 7 – Графическое представление SQL модели данных.

3.2.2. Описание назначений коллекций, типов данных, сущностей.

В реляционной модели данных необходимо 3 таблицы – Symptom, содержащая id и название симптомов, Diagnosis, содержащая id и название диагноза, а также общая таблица Diagnostics с соответствием – какие симптомы являются признаками каких болезней. Таблицы Symptom и Diagnostic, так же, как и Diagnosis и Diagnostics, связаны отношением «один ко многим».

3.2.3. Оценка удельного объема информации, хранимой в модели.

Пусть n – количество записей в Symptom, а m – количество записей в Diagnosis, следовательно, количество записей в Diagnostics будет не меньше $n \cdot m$. Удельный объем информации получается $V \approx 260n + 260m + 256nm$

3.2.4. Запросы к модели, с помощью которых реализуются сценарии использования.

а) Узнать болезнь

Количество требующихся запросов равно 6:

- Поиск симптомов
- Поиск индексов диагнозов, содержащие выбранные пользователем симптомы.
- Поиск названия диагнозов.
- Поиск общего количества симптомов для каждого диагноза
- Создание и заполнение итоговой таблицы с диагнозами и их вероятностью

```
Q1
SELECT syd, count(did) a kol from Diagnosis
GROUP BY syd
ORDER BY DESC
LIMIT 10;

Q2
SELECT did FROM Diagnosis
WHERE syd = (SELECT syd FROM Q1);

Q3
SELECT did, Count (syd) as kol from Diagnosis
GROUP BY did;

Q4
SELECT symptom FROM Symptom
WHERE id = (SELECT syd FROM Q1)

Q4
SELECT diagnosis FROM Diagnoses
WHERE id = (SELECT did FROM Q2)

Q5
CREATE TABLE Result (
id INT NOT NULL
Dia VARCHAR(255)
rate INT);

Q6
```

```
INSERT INTO Result (id, Dia,rate)
VALUES(i,"Headache","ratevalue"
```

б) Построение столбчатой диаграммы для выбранных симптомов.

Аналогично пункту (а) требуется 6 запросов.

в) Наиболее часто встречающиеся симптомы.

Требуется 2 запроса:

- Поиск количества диагнозов, соответствующих каждому симптому и распределение их по убыванию этого количества.
- Поиск названия симптомов

г) Наиболее трудно диагностируемые диагнозы.

Требующееся количество запросов равно 6.

- Поиск количества диагнозов, соответствующих каждому симптому, и распределение их по убыванию этого количества.
- Поиск диагнозов, имеющие часто встречающиеся симптомы.
- Поиск общего количества болезней для каждого диагноза, имеющие часто встречающиеся симптомы.
- Поиск названия симптомов.
- Поиск названия диагнозов.
- Создание и заполнение итоговой таблицы с диагнозами и их вероятностью.

д) Среднее число симптомов для определения диагноза.

В данном случае требуется 2 запроса:

- Поиск общего количества симптомов для каждого диагноза.
- Поиск количества записей диагнозов.

3.3. Сравнение моделей

NoSQL модель дает выигрыш в используемой памяти, так как функция издержек определяется константой, и это решение позволяет создания минимального размера запроса, в то время как SQL решение - напротив нерационально использует память (издержки растут с

квадратичной скоростью) и размер запросов заметно больше. Проблема дублирования информации в neo4j решается за счет использования графового представления данных, что позволяет сократить место на диске, кроме того, данное решение легко реализуется.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание

Разработанное приложение предназначено для хранения и анализа медицинских данных. Пользователь имеет возможность узнать вероятность болезней, выбрав собственные симптомы из списка. Кроме того, приложение предоставляет возможность получения статистики по используемым данным, а именно, список самых размытых симптомов, список самых трудно диагностируемых болезней, а также среднее количество симптомов для каждой болезни.

Кроме того, пользователь может экспортировать данные из приложения, а также загрузить в приложение свои собственные данные.

Хранение и управление данными в приложении реализовано при помощи Neo4j.

4.2. Схема экранов приложения

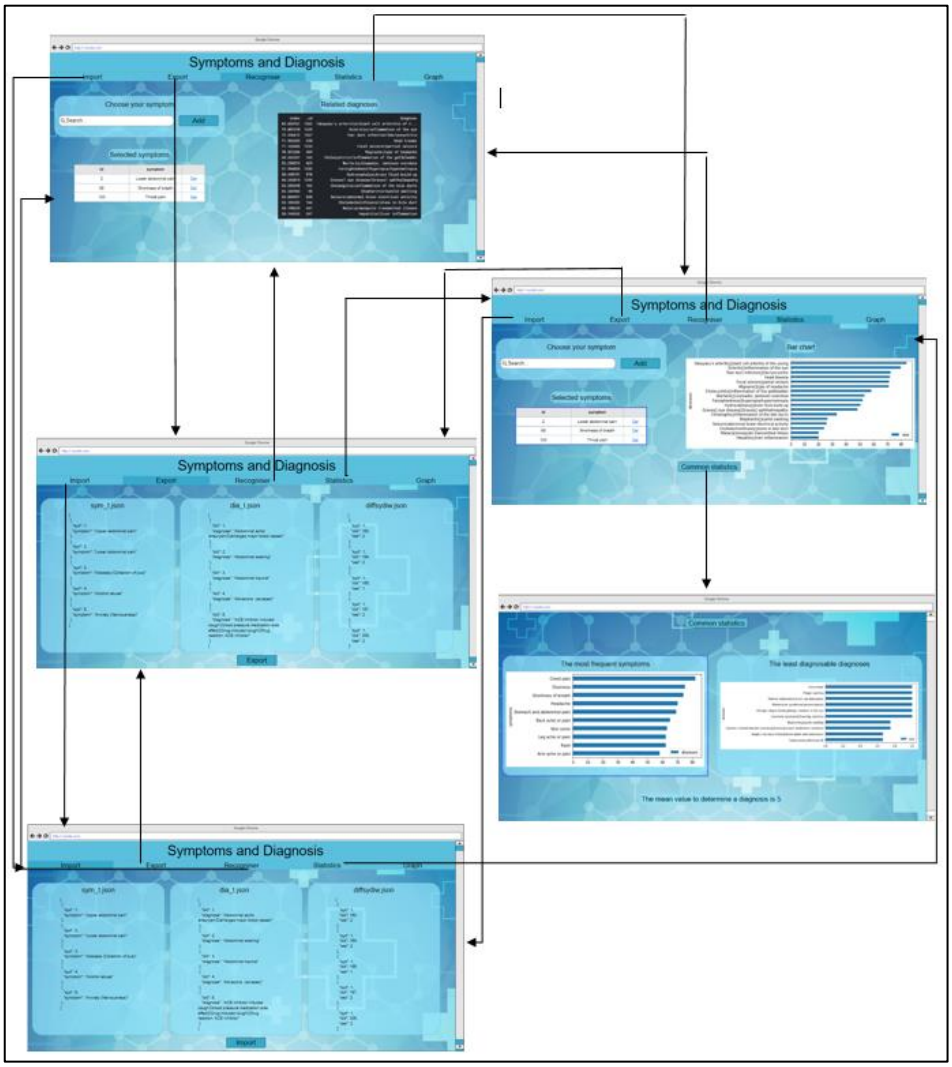

Рисунок 8 – Схема экранов приложения.

4.3. Использованные технологии

Приложение написано на языке Python 3 с использованием библиотеки py2neo для использования Neo4j, а также библиотек matplotlib, numpy, pandas, PIL, tkinter.

4.4. Ссылки на приложение

Ссылка на разработанное приложение приведена в разделе «Список использованных источников» [1]

ЗАКЛЮЧЕНИЕ

В результате выполнения работы с использованием СУБД Neo4j было реализовано приложение, в котором можно получить информацию о возможном заболевании по выбранным симптомам. Реализованы функции импорта и экспорта данных, а также имеется возможность просмотра статистики по симптомам и болезням. Таким образом, поставленная цель была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Исходный код приложения: https://github.com/moevm/nosql2h19-med
2. Документация Neo4j: https://neo4j.com/docs/
3. Документация py2neo: https://py2neo.org/2.0/
4. Документация matplotlib: https://matplotlib.org/3.1.1/contents.html
5. Введение в нереляционные базы данных: https://stepik.org/course/2586/syllabus