Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ (ИУ)

КАФЕДРА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Разработка базы данных карточек для адаптивной коммуникации

Студент группы **ИУ7-66Б** _____ **Авсюнин А. А.**

Руководитель курсовой работы _____ **Строганов Ю. В.**

2023 г.

РЕФЕРАТ

Рассчётно-пояснительная записка содержит 29 с., 9 рис., 2 табл., 13 ист.

Ключевые слова: базы данных, СУБД, нереляционная модель, Neo4j, адаптивная коммуникация, карточки PECS.

Цель работы: создание базы данных карточек для адаптивной коммуникации.

В данной работе изучаются принципы работы с базами данных, разрабатывается и реализуется приложение для адаптивного общения карточками.

Результаты: программный продукт, обеспечивающий взаимодействие с графовой базой данных.

Содержание

Введение

Адаптивное общение — это форма общения, которая адаптирована к чьим-либо потребностям и способностям. Она предназначена для предоставления людям возможности общаться с другими людьми, даже если они не могут участвовать в разговорной речи. Одним из способов реализации такого общения являются карточки со словами, обозначающими предмет или действие.

Целью данной работы является создание базы данных карточек для адаптивной коммуникации. Для достижения поставленной цели необходимо выполнить следующие задачи:

- проанализировать существующие СУБД;
- описать сущности проектируемой БД;
- выбрать необходимый инструментарий для реализации;
- реализовать спроектированную БД и необходимый интерфейс для работы с ней;
- исследовать характеристики разработанного программного обеспечения.

1 Аналитический раздел

В данном разделе проводится анализ предметной области, описание карточек для альтернативного взаимодействия PECS, проводится классификация существующих СУБД. На основе анализа предметной области составляется список пользователей системы, а также создаётся диаграмма вариантов использования, а также диаграмма сущность-связь в нотации Чена. На основе данных, которые будут хранится в системе выбирается модель базы данных.

1.1 Понятие базы данных

Определение БД

Большинство определений понятия «база данных» носят субъективный характер и мнение тех или иных авторов. Однако можно отметить наиболее общие из них.

База данных [1] — это некоторый набор перманентных (постоянно хранимых) данных, используемых прикладными программными системами какого-либо предприятия.

База данных — это самодокументированное собрание интегрированных записей.

Типы баз данных

По типу применения базы данных делятся на:

1. **OLAP**(online analytical processing) — это технология обработки данных, которая используется для анализа больших объемов данных.

2. **OLTP**(online transactional processing) — это технология обработки транзакций, которая используется для выполнения операций в режиме реального времени.

В силу необхоимости работать в режиме реального времени для разрабатываемого программного продукта будет выбрана технология OLTP.

1.2 Понятие системы управления базами данных

Определение СУБД

Для взаимодействия с любой базой данных необходима система управления.

Система управления базами данных (СУБД) — это приложение обеспечивающее создание, хранение, обновление и поиск информации в базе данных.

Классификация СУБД

По модели данных

1. **Дореляционные**

   a. **Инвертированные списки** (файлы). БД на основе инвертированных списков представляет собой совокупность файлов, содержащих записи (таблиц). Для записей в файле определен некоторый порядок, диктуемый физической организацией данных. Для каждого файла может быть определено произвольное число других упорядочений на основании значений некоторых полей записей (инвертированных списков). Обычно для этого используются индексы. В такой модели данных отсутствуют ограничения целостности как таковые. Все ограничения на возможные экземпляры БД задаются теми программами, которые работают с БД. Одно из немногих ограничений, которое все-таки может присутствовать — это ограничение, задаваемое уникальным индексом.

   b. **Иерархическая модель** данных подразумевает что элементы, организованные в структуры, объединены иерархической или древовидной связью. В таком представлении родительский элемент может иметь несколько дочерних, а дочерний — только один родительский.

   c. **Сетевые** — могут быть представлены в виде графа; логика выборки зависит от физической организации данных.

2. **Реляционные**

В отличие от дореляционных, в данной модели не существует физических отношений между сущностями. Хранение информации осуществляется в виде таблиц (отношений), состоящих из рядов и столбцов. Отношение имеет имя, которое отличает его от имён всех других отношений.

a. **Структурные** — данные — набор отношений.

b. **Целостностные** — отношения (таблицы) отвечают определенным условиям целостности.

c. **Манипуляционные** — манипулирования отношениями осуществляется средствами реляционной алгебры и/или реляционного исчисления.

3. **постреляционные [2]**

Основным недостатком реляционных баз данных является ограничение поддерживаемых типов. Большинство постреляционных баз данных основаны на хранении данных в виде сложных структур.

a. **Объектные** — данные моделируются в виде объектов;

b. **Объектно-реляционные** — позволяет использовать возможности объектно-ориентированного подхода: объекты, классы, наследование.

По архитектуре организации хранения данных

1. **Локальные** — все части локальной СУБД размещаются на одном компьютере.

2. **Распределенные** — части СУБД могут размещаться на 2-х и более компьютерах.

По способу доступа к БД

1. **Файл-серверные** — при работе с базой, данные отправляются приложению, которое с ней работает, вне зависимости от того, сколько их нужно. Все операции — на стороне клиента. Файловый сервер периодически обновляется тем же клиентом.

2. **Клиент-серверные** — вся работа происходит на сервере, по сети передаются результаты запросов, гораздо меньше информации. Обеспечивается безопасность данных, так как все действия происходят на стороне сервера.

3. **Встраиваемые** — библиотека, которая позволяет унифицированным образом хранить большие объемы данных на локальной машине. Доступ к данным может происходить через SQL либо через особые функции СУБД. Встраиваемые СУБД быстрее обычных клиент-серверных и не требуют установки сервера, поэтому востребованы в локальном ПО, которое имеет дело с большими объемами данных.

4. **Сервисно-ориентированные** — БД является хранилищем сообщений, промежуточных состояний, метаинформации об очередях сообщений и сервисах;

5. **Прочие** — пространственная, временная и пространственно-временная.

1.3 Карточки PECS

PECS [3] – это система альтернативной коммуникации, где общение происходит методом обмена изображениями с помощью специальных карточек. Они позволяют заменить или дополнить обычную речь человеку, который из-за своего заболевания не способен объяснять понятно для большинства людей.

Каждая карточка содержит слово и картинку, описывающую данное слово. Их использование в обучении детей с аутизмом способствует развитию речи, помогает развивать абстрактное мышление, улучшает понимание вербальной речи, позволяет улучшить качество жизни человека и повысить уровень его социализации. Примеры карточек приведены на рисунке 1.1.

Программа PECS разработана в прошлом веке в конце 80-х годов доктором Э. Бонди в тесном сотрудничестве с логопедом Л. Фрост. После появления первой версии программа постоянно совершенствуется. Сегодня система PECS приспособлена для обучения не только детей с аутизмом, но также пациентов любых возрастных категорий с коммуникационными и речевыми проблемами.

Рисунок 1.1 – Примеры карточек PECS

1.4 Пользователи системы

В системе должно быть три уровня пользователей:

1. Психолог — пользователь, обладающий возможностями изменять, добавлять, удалять и просматривать сущности базы данных;
2. Пациент — пользователь, обладающий возможностью только просматривать сущности базы данных.
3. Родственник — пользователь, обладающий возможностями изменять и просматривать сущности базы данных;

На рисунке 1.2 представлена диаграмма вариантов использования.

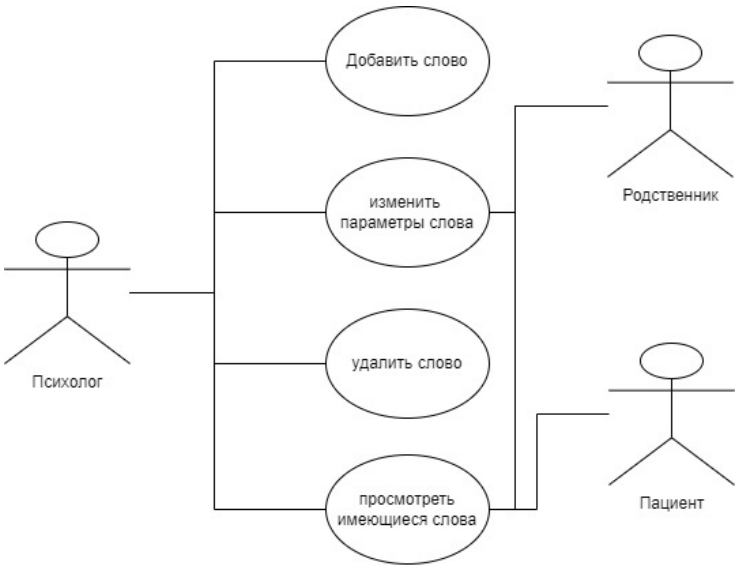

Рисунок 1.2 – Диаграмма вариантов использования

Основной пользователь приложения, то есть пользователь с ролью «Пациент» может использовать базу данных для разных целей:

1. составление расписания — пользователю необходимо чётко обозначать план действий на ближайший день, что реализуется с помощью последовательности из карточек PECS;

2. составление истории — пользователь использует карточки PECS для составления рассказа, несколько карточек помещаются на одну большую карту, обозначающую место действия, вместе карточки образуют непрерывную историю;

3. взаимодействие с глобальной сетью Интернет — пользователь с помощью карточек PECS создаёт запрос в сеть, который обрабатывается и преобразуется в ответ поисковой системы, представленный также карточками PECS;

1.5 Описание данных

База данных будет состоять из 5 таблиц:

1. таблица владельцев словарей Owner;

2. таблица главных слов «Термин»;

3. таблица действий субъекта «Действие субъекта»;

4. таблица действий объекта «Действие объекта»;

5. таблица характеристик «Характеристика».

Каждая таблица описывает сущность базы данных, к которым будут прилагаться поля, хранящие в себе информацию о данной сущности.

На рисунке 1.3 представлена ER-диаграмма в нотации Чена.

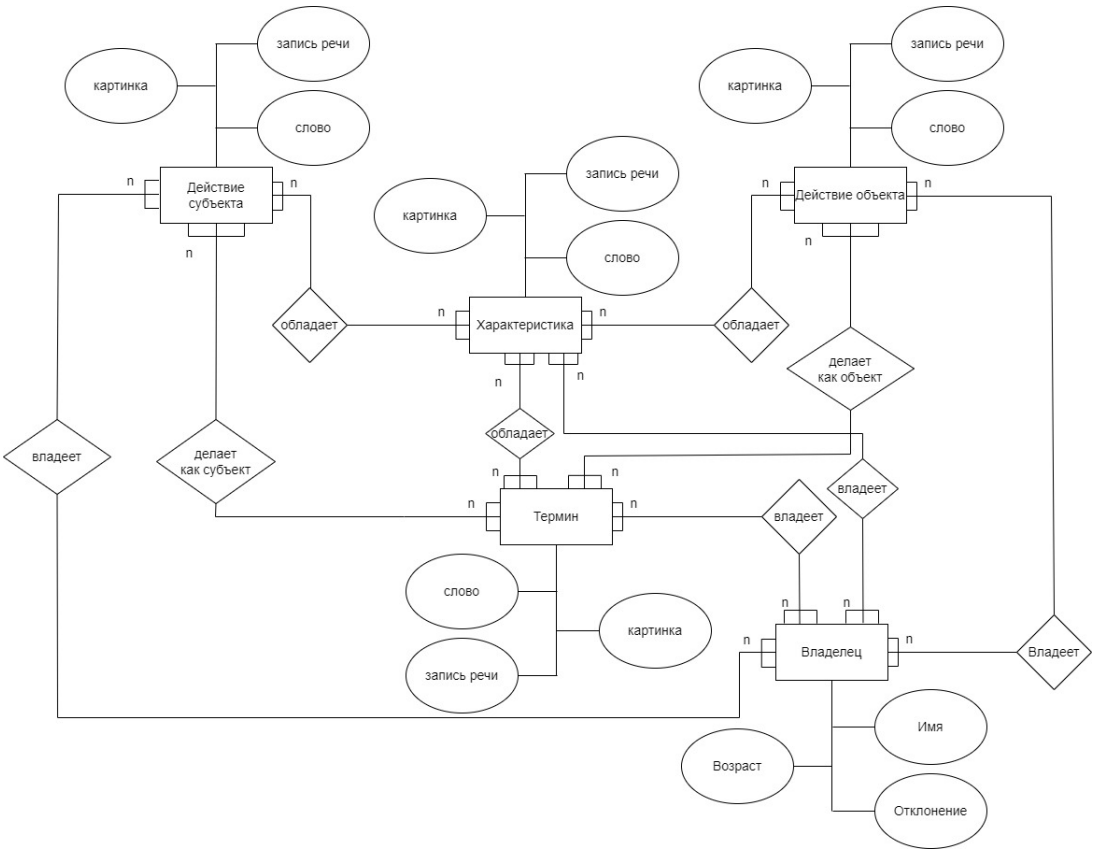

Рисунок 1.3 – ER-диаграмма в нотации Чена

1.6 Выбор модели БД

Основываясь на представлении данных в данной работе будет использоваться графовая нереляционная модель. Такой подход обладает следующими характеристиками:

1. отсутствие развязочных таблиц;
2. произвольный доступ к данным;
3. исключение дублирования за счёт множественных связей.

Вывод

В данном разделе был проведён анализ предметной области, описаны карточки для адаптивного общения, а также приведена классификация существующих СУБД. Выбраны пользователи системы, описан вид хранения данных и приведены соответствующие диаграммы.

2 Конструкторский раздел

В данном разделе формализуются сущности системы, определяется ролевая модель и выбираются необходимые процедуры/функции/триггеры. Каждой сущности, описанной в аналитическом разделе соотносятся поля — информация, необходимая для полного их определения. На основе этой информации создаётся ER-диаграмма базы данных, в которой указываются все таблицы необходимы для правильной работы системы. Также создаётся схема триггера, используемого в системе.

2.1 Описание сущностей

В предыдущем разделе были сформулированы сущности разрабатываемой базы данных. Далее приведён полный набор полей каждой сущности.

1. **Owner** — таблица владельцев словарей;
    a. id — порядковый номер владельца, число, первичный ключ;
    b. name — имя владельца, строка;
    c. age — дата рождения владельца, дата;
    d. disease — отклонение владельца, строка;
2. **Noun** — главные слова;
    a. id — порядковый номер слова, число, первичный ключ;
    b. ownerid — порядковый номер владельца, число, внешний ключ;
    c. word — само слово, строка;
    d. picture — картинка PECS, фотография;
    e. record — запись произношения слова, звуковой файл;
3. **SubAction** — действия термина как субъекта;
    a. id — порядковый номер слова, число, первичный ключ;
    b. word — само слово, строка;
    c. picture — картинка PECS, фотография;

d. record — запись произношения слова, звуковой файл;

4. **ObAction** — действие термина как объекта;

   a. id — порядковый номер слова, число, первичный ключ;

   b. word — само слово, строка;

   c. picture — картинка PECS, фотография;

   d. record — запись произношения слова, звуковой файл;

5. **Adjective** — характеристики слов;

   a. id — порядковый номер слова, число, первичный ключ;

   b. word — само слово, строка;

   c. picture — картинка PECS, фотография;

   d. record — запись произношения слова, звуковой файл.

На рисунке 2.1 представлена ER-диаграмма базы данных.

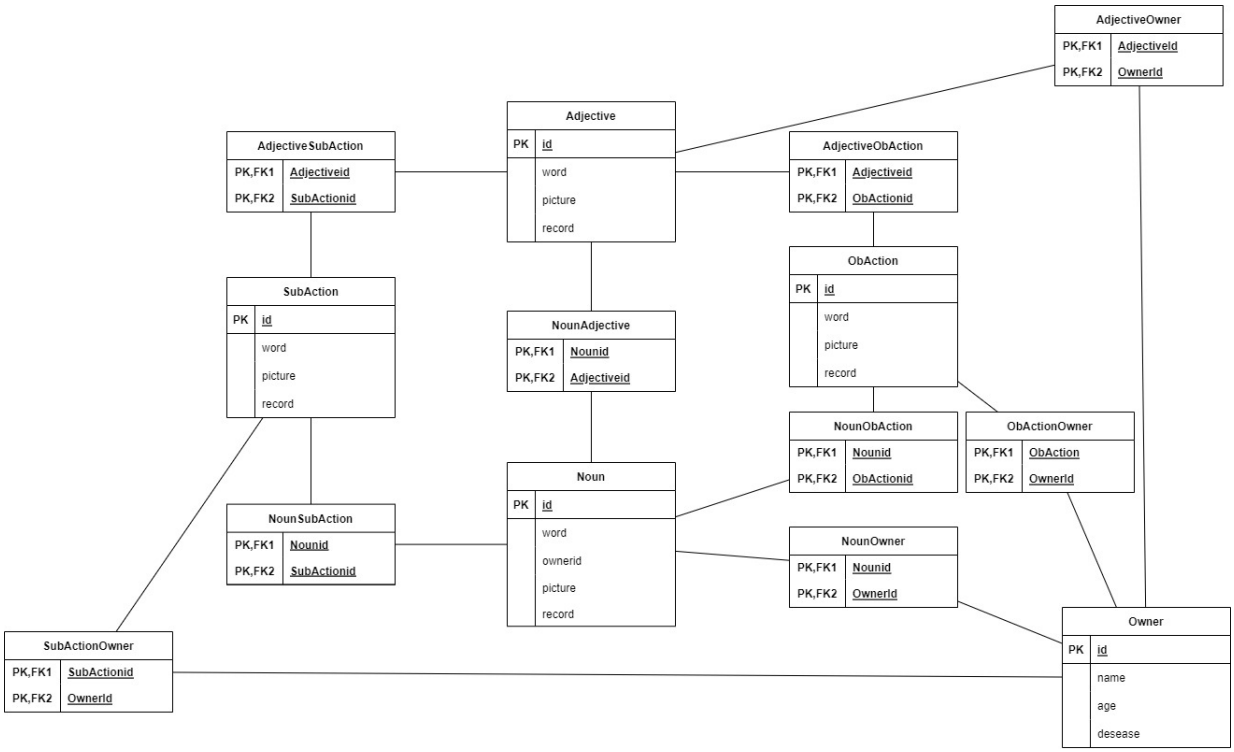

Рисунок 2.1 – ER-диаграмма базы данных

2.2 Ролевая модель

Ролевая модель в системе необходима для того, чтобы правильно организовать работу пользователей в системе, предоставляя каждому пользователю определённый набор действий в системе.

Для работы пользователей с базой данных выделяется следующая ролевая модель:

1. doctor — психолог. Имеет все права над всеми таблицами;
2. patient — пациент. Имеет доступ SELECT ко всем таблицам.
3. family — родственник. Имеет доступ SELECT и UPDATE ко всем таблицам.

2.3 Используемые триггеры

В системе представлен триггер, позволяющий удалять все узлы, оказавшиеся без единой связи с сущностью «Владелец», после удаления очередного узла из БД.

На рисунке 2.2 представлена схема триггера.

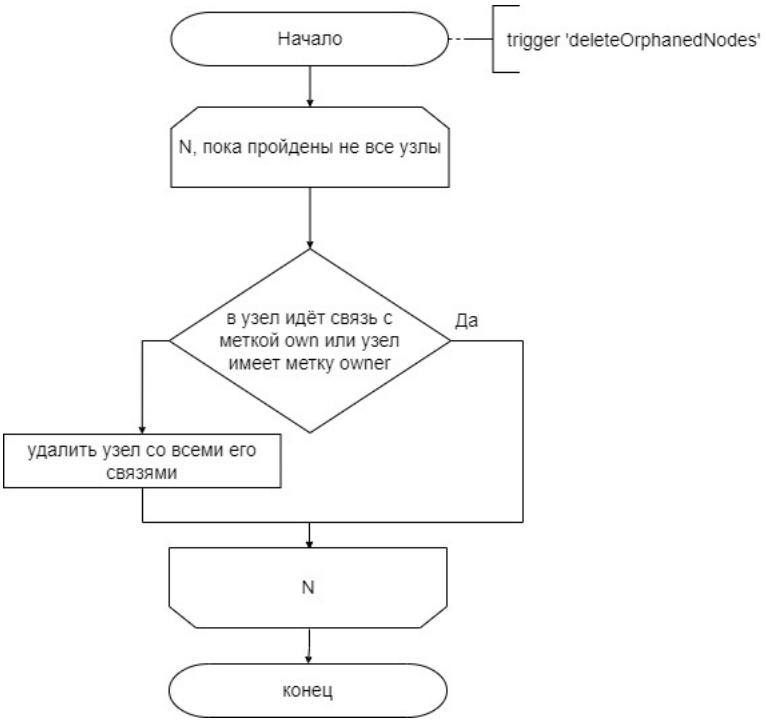

Рисунок 2.2 – Схема триггера

Вывод

В данном разделе были целиком описаны сущности системы, представлена ER-диаграмма базы данных, выбрана ролевая модель. Описаны используемые процедуры/функции/триггеры.

3 Технологический раздел

В данном разделе делается выбор используемой СУБД и других средств реализации приложения, создаются триггер и ролевая модель в системе. Описывается создание пользовательского интерфейса. Демонстрируется работа созданного приложения.

Ролевая модель создаётся на основе той, что была сформулирована в аналитическом разделе. Представляется код на языке Cypher для создания ролей, а также триггера.

3.1 Выбор СУБД

Самыми распространёнными [4] СУБД графового типа являются JanusGraph [5], Neo4j [6] и Dgraph [7].

1. **JanusGraph**. Встроенный сложный поиск, а также дополнительная (опциональная) поддержка Elasticsearch. Использование полного по Тьюрингу языка запросов для обхода графов.

2. **Neo4j**. Масштабируемость БД за счет разделения данных на части – сегменты. Высокий уровень безопасности: несколько экземпляров баз данных можно разделить, оставив их на одном выделенном сервере. Использование Cypher – языка запросов для графовых БД.

3. **Dgraph**. Горизонтальная масштабируемость для работы в реальной среде с ACID-транзакциями. Использование языка запросов GraphQL.

В данной работе используется Neo4j, так как эта СУБД предоставляет достаточный инструментарий для реализации поставленных задач.

3.2 Выбор средств реализации

В качестве языка программирования для реализации программного продукта был выбран Python [8]. Для взаимодействия с СУБД используется библиотека neo4j [9], которая, как и другие python-библиотеки, устанавливается с помощью встроенного в python пакетного менеджера pip. Также в

СУБД Neo4j используется дополнительный плагин APOC [10], предоставляющий необходимый набор процедур для работы с Neo4j, одной из которых является процеудра регистрации нового триггера.

3.3 Создание базы данных

В данной работе требуется задать ролевую модель на основе разработанной в аналитическом разделе и создать удаляющий триггер.

Создание ролевой модели

В конструкторском разделе была разработана следующая ролевая модель:

1. doctor — психолог. Имеет все права над всеми таблицами;
2. patient — пациент. Имеет доступ SELECT ко всем таблицам;
3. family — родственник. Имеет доступ SELECT и UPDATE ко всем таблицам.

Сценарий создания ролей представлен в Листинге 3.1.

Листинг 3.1 – Создание ролевой модели

```
1 CREATE ROLE doctor
2 CREATE ROLE patient
3 CREATE ROLE family
4 GRANT MATCH {*} ON GRAPH dictionary to doctor
5 GRANT WRITE ON GRAPH dictionary to doctor
6 GRANT MATCH {*} ON GRAPH dictionary to patient
7 GRANT MATCH {*} ON GRAPH dictionary to family
8 GRANT SET PROPERTY {*} ON GRAPH dictionary to family
9 GRANT SET LABEL {*} ON GRAPH dictionary to family
```

Создание триггера

Сценарий создания триггера представлен в Листинге 3.2.

Листинг 3.2 – Создание триггера

```
1 CALL apoc.trigger.add('deleteOrphanedNodes', 'UNWIND $deletedNodes AS dn
    MATCH (n) WHERE NOT ()-[:own]->(n) AND NOT n:owner DETACH DELETE n' ,
    {phase: 'after'})
```

3.4 Интерфейс взаимодействия

Для работы с БД создан API с помощью библиотеки fastapi [11]. В программном интерфейсе реализованы методы для выполнения операций создания, чтения, обновления и удаления узлов в базе данных. На рисунке 3.1 представлена часть имеющихся методов, таких как:

1. создать:
    a. термин;
    b. действие объекта;
    c. действие субъекта;
2. получить:
    a. все узлы;
    b. все термины владельца;
    c. все термины;
3. изменить:
    a. термин;
    b. действие объекта;
    c. декйствие субъекта;
4. удалить:
    a. термин;
    b. действие объекта;
    c. действие субъекта.

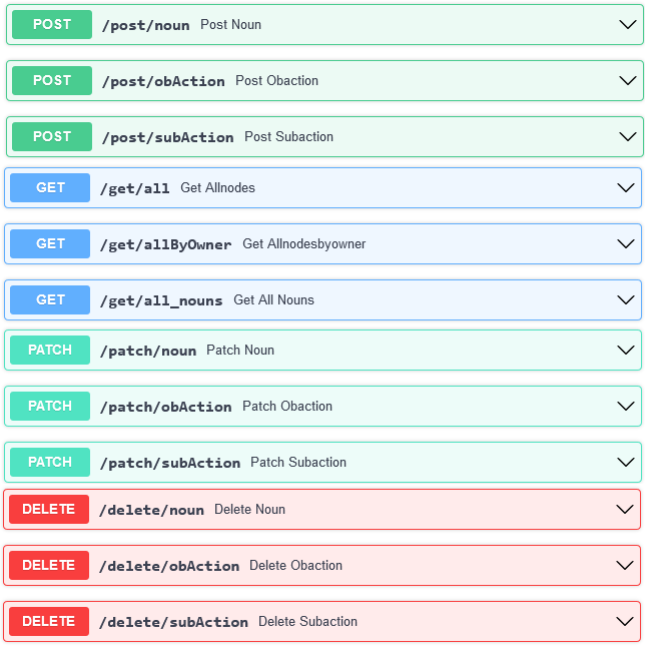

Рисунок 3.1 – Программный интерфейс

3.5 Демонстрация работы

На рисунке 3.2 демонстрируется процесс получения ответа на запрос о получении всех возможных узлов в базе данных. Данный запрос требует на вход единственный параметр — роль. Результатом запроса является JSON объект с параметрами status — состояние запроса, results — количество полученных узлов и nodes — сами узлы. Результат отображён в нижней части рисунка 3.2.

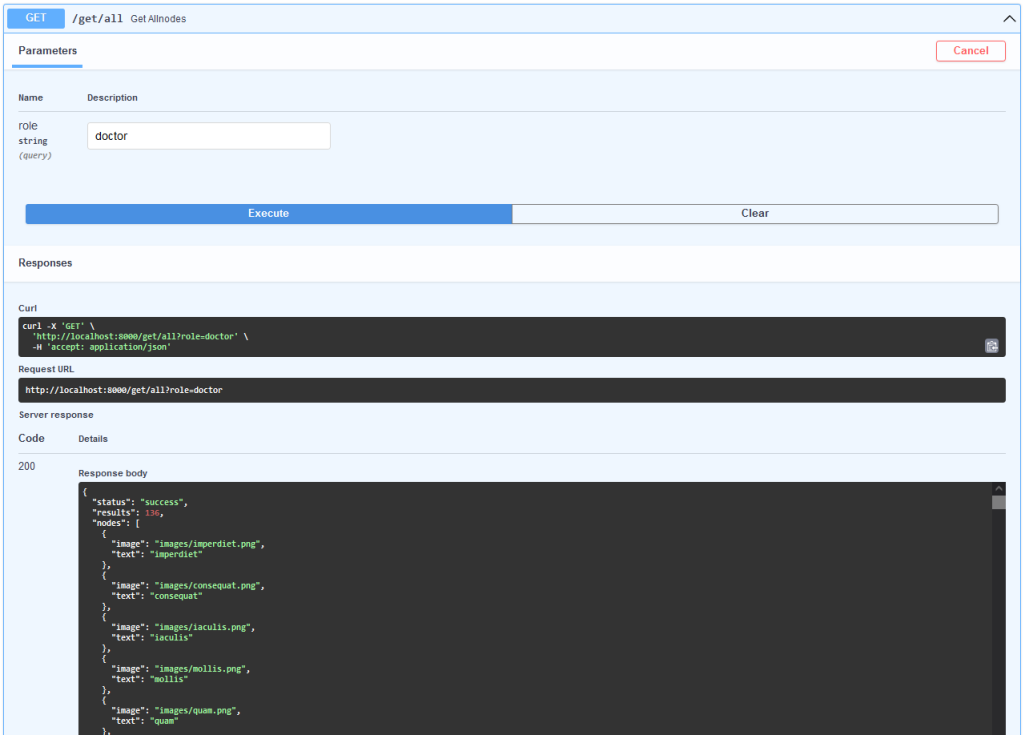

Рисунок 3.2 – Демонстрация работы программы

Вывод

В данном разделе выбраны СУБД и средства реализации, представлено создание ролевой модели и триггера. Описан пользовательский интерфейс.

4 Исследовательский раздел

В данном разделе проводится исследование характеристик созданной базы данных. А именно исследование времени поиска и создания узла. Перед проведением исследования указываются технические характеристики устройства, на котором данное исследование проводится. На основе полученных результатов делается вывод о том, как на исследуемые характеристики системы зависят от количества существующих узлов в базе данных.

4.1 Технические характеристики

При проведении исследования какой-либо системы важно знать какими характеристиками обладает устройство, на котором проводятся измерения, так как численные результаты могут изменяться от устройства к устройству. Технические характеристики устройства, на котором выполнялись измерения:

1. операционная система: Windows 10 Pro Версия 21H1 x64 [12];
2. оперативная память: 8 ГБ;
3. процессор: 2,4 ГГц 4-ядерный процессор Intel Core i5-1135G7[13].

Во время замеров ноутбук был включен в сеть электропитания, нагружен только самим приложением.

4.2 Анализ времени поиска узла по связям

В процессе использования приложения пациент будет составлять предложения из слов, возможна ситуация в которой необходимо соединить 2 слова неким другим словом, связанным с начальными. В такой ситуации требуется как можно быстрее находить все варианты недостающего слова в базе данных.

Исследование проводится для получения зависимости времени получения ответа от количества узлов в базе данных.

Таблица 4.1 – Замеры времени
поиска узла по двум «соседним»

Количество узлов	Время (мс)
15	1
25	2
40	3.5
50	3.5
70	5.5
90	7
100	8
115	8.5
135	9

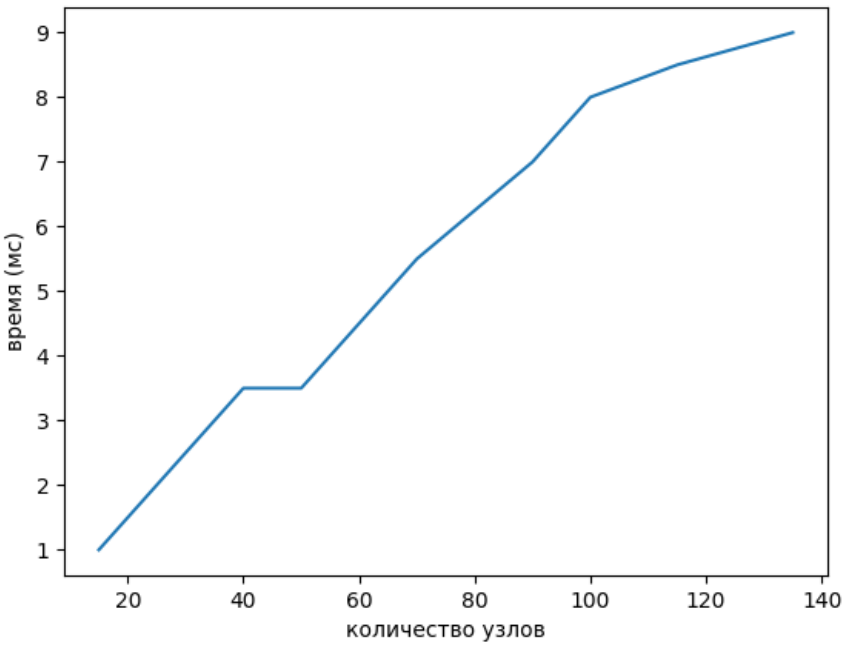

Рисунок 4.1 – Замеры времени поиска узла по двум «соседним»

В таблице 4.1 и на графике 4.1 представлены результаты замеров времени при разном количестве узлов.

По результатам замеров видно, что время линейно зависит от количества узлов в базе данных.

4.3 Анализ времени добавления узла

В процессе развития пациента в базу данных будут добавляться новые слова.

Исследование проводится для получения зависимости времени добавления узла от количества уже имеющихся узлов в базе.

В таблице 4.2 и на графике 4.1 представлены результаты замеров времени при разном количестве узлов.

Таблица 4.2 – Выборка из
замеро времени добавления узла

Количество узлов	Время (мс)
50	2.8
100	2.0
150	1.0
200	2.5
250	3.8
300	3.7
350	3.4
400	5.5
450	3.7
500	3.1

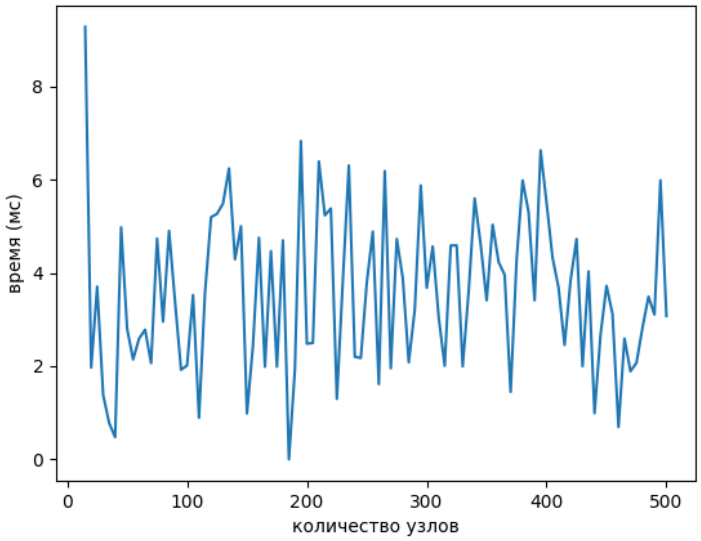

Рисунок 4.2 – Замеры времени добавления узла

По результатам замеров видно, что время создания нового узла не зависит от времени.

Вывод

В данном разделе были представлены технические характеристики устройства на котором проводились измерения. В процессе исследования было обнаружено, что время поиска узла по двум «соседним» линейно завит от количества узлов в базе данных. Однако также замечено, что время создания узла не зависит от общего количества узлов.

Заключение

Цель, поставленная в начале, была достигнута: разработана база данных карточек для адаптивной коммуникации. В ходе выполнения курсовой работы были решены следующие задачи:

- проанализированы существующие СУБД;
- описаны сущности проектируемой БД;
- выбран необходимый инструментарий для реализации;
- реализованы спроектированная БД и необходимый интерфейс для работы с ней;
- исследованы характеристики разработанного программного обеспечения.

Список использованных источников

1. К.Дж. Дейт. Введение в системы баз данных. — Москва: издательский дом «Вильяме», 2005. — С. 51.

2. В.И. Виноградов. Постреляционные модели баз данных и языки запросов: учеб.пособие. — Москва: Изд-во МГТУ им. Н.Э. Баумана, 2016.

3. Карточки PECS для адаптивной коммуникации. [Электронный ресурс]. Режим доступа: `https://chichiko.ru/blog/kartochki-pecs-dlya-detey-s-autizmom/` (дата обращения: 10.04.2022).

4. Что такое графовая база данных [Электронный ресурс]. Режим доступа: `https://wiki.merionet.ru/servernye-resheniya/101/chto-takoe-grafovaya-baza-dannyh/` (дата обращения: 20.04.2022).

5. JanusGraph [Электронный ресурс]. Режим доступа: `https://janusgraph.org/` (дата обращения: 20.04.2022).

6. Neo4j [Электронный ресурс]. Режим доступа: `https://neo4j.com/` (дата обращения: 20.04.2022).

7. Dgraph [Электронный ресурс]. Режим доступа: `https://dgraph.io/` (дата обращения: 20.04.2022).

8. Python [Электронный ресурс]. Режим доступа: `https://python.org/` (дата обращения: 20.04.2022).

9. Using Neo4j from Python [Электронный ресурс]. Режим доступа: `https://neo4j.com/developer/python/` (дата обращения: 20.04.2022).

10. Neo4j APOC Library [Электронный ресурс]. Режим доступа: `https://neo4j.com/developer/neo4j-apoc/` (дата обращения: 20.04.2022).

11. FastAPI [Электронный ресурс]. Режим доступа: `https://fastapi.tiangolo.com/` (дата обращения: 20.04.2022).

12. Windows [Электронный ресурс]. Режим доступа: `https://www.microsoft.com/ru-ru/windows` (дата обращения: 20.04.2022).

13. Процессор Intel® Core™ i5 [Электронный ресурс]. Режим доступа: https://www.intel.com/processors/core/i5/docs (дата обращения: 20.04.2022).

ПРИЛОЖЕНИЕ А

<слайды презентации>