

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: Сервис составления генеалогических деревьев

Студент гр. 0303		Пичугин М.В.
Студент гр. 0303	_____	Середенков А.А.
Студент гр. 0303	_____	Сологуб Н.А.
Преподаватель	_____	Заславский М.М.

Санкт-Петербург
2023

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студенты

Пичугин М.В.

Середенков А.А.

Сологуб Н.А.

Группа 0303

Тема проекта: Разработка приложения для составления генеалогических деревьев.

Исходные данные:

Необходимо реализовать приложение для составления генеалогических деревьев с помощью СУБД Neo4j.

Содержание пояснительной записки:

«Содержание», «Введение», «Сценарий использования», «Модель данных», «Разработанное приложение», «Выводы», «Приложения», «Литература»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 20.09.2023

Дата сдачи реферата: 20.12.2023

Дата защиты реферата: 20.12.2023

Студент гр. 0303		Пичугин М.В.
Студент гр. 0303		Середенков А.А.
Студент гр. 0303		Сологуб Н.А.
Преподаватель		Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема создания приложения для создания генеалогических деревьев с помощью СУБД Neo4j, так как это отличная возможность понять принцип работы графовых баз данных для данной предметной области. Во внимание будут приниматься такие аспекты как производительность и удобство разработки. Найти исходный код и всю дополнительную информацию можно по ссылке: <https://github.com/moevm/nosql2h23-genealogy>

ANNOTATION

As part of this course, it is planned to prepare an application in a team on one of the assigned topics. The topic of creating an application for creating family trees using the Neo4j DBMS was chosen, as this is an excellent opportunity to understand the principle of operation of graph databases for this subject area. Aspects such as performance and ease of development will be taken into account. You can find the source code and all additional information at the link: <https://github.com/moevm/nosql2h23-genealogy>

СОДЕРЖАНИЕ

ЗАДАНИЕ.....	2
АННОТАЦИЯ.....	3
СОДЕРЖАНИЕ.....	4
1. ВВЕДЕНИЕ.....	5
1.1. Актуальность решаемой проблемы	6
1.2. Постановка задачи	6
1.3. Предлагаемое решение	6
1.4. Качественные требования к решению	6
2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ.....	7
2.1. Макеты UI.....	7
2.2. Сценарии использования для задачи	10
2.2.1. Импорт данных	10
2.2.1.1. Сценарий использования - “массовый импорт данных”.....	10
2.2.1.2. Сценарий использования - “ручной импорт данных”	11
2.2.2. Представление данных	11
2.2.3. Анализ данных	12
2.2.4. Экспорт данных	12
2.2. Вывод относительно нашего функционала	13
3. МОДЕЛЬ ДАННЫХ.....	14
3.1. Нереляционная модель данных - Neo4j	14
3.1.1. Графическое представление данных	14
3.1.2. Описание назначений коллекций, типов данных и сущностей	14
3.1.3. Оценка удельного объема информации, хранимой в модели	15
3.1.4. Запросы к модели, с помощью которых реализуются сценарии использования	16
3.2. Реляционные модели данных.....	17
3.2.1. Графическое представление данных.....	17
3.2.2. Описание назначений коллекций, типов данных и сущностей.....	18
3.2.3. Оценка удельного объема информации, хранимой в модели.....	19

3.2.4. Запросы к модели, с помощью которых реализуются сценарии использования.....	20
3.3. Сравнение моделей.....	21
3.3.1. Удельный объем информации	21
3.3.2. Запросы по отдельным юзкейсам	21
3.3.3. Вывод.....	21
4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ.....	22
4.1. Краткое описание.....	22
4.2. Используемые технологии.....	22
4.3. Снимки экрана приложения.....	22
5. ВЫВОД.....	27
5.1. Достигнутые результаты.....	27
5.2. Недостатки и пути для улучшения полученного решения.....	27
5.3. Будущее развитие решения.....	27
6. ПРИЛОЖЕНИЯ.....	28
6.1. Документация по сборке и развертыванию приложения.....	28
6.2. Инструкция для пользователя.....	28
7. ЛИТЕРАТУРА.....	29

1. ВВЕДЕНИЕ

1.1 Актуальность решаемой проблемы

Цель работы - создать быстрое и удобное веб-приложения по созданию, хранению и обработке генеалогических деревьев пользователей.

1.2 Постановка задачи

Сервис по составлению генеалогических деревьев должен решать следующие задачи:

- Позволять пользователям создавать и редактировать генеалогические деревья.
- Предоставлять пользователям инструменты для сбора и хранения информации о своих предках.

1.3 Предлагаемое решение

Для решения поставленной задачи необходимо разработать web приложение по составлению генеалогических деревьев.

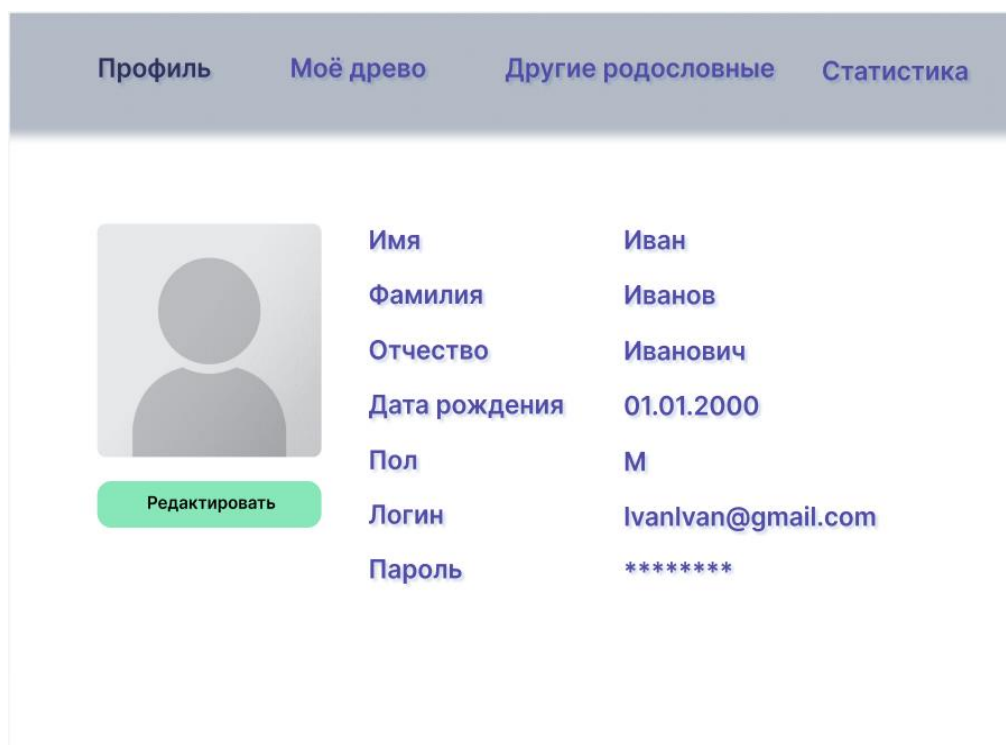
1.4 Качественные требования к решению

Требуется разработать веб-приложение с использованием СУБД neo4j и фреймворков Vue.js и Express.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макеты UI

1. Экран профиля (Рис. 1)



Профиль	Моё древо	Другие родословные	Статистика
---------	-----------	--------------------	------------



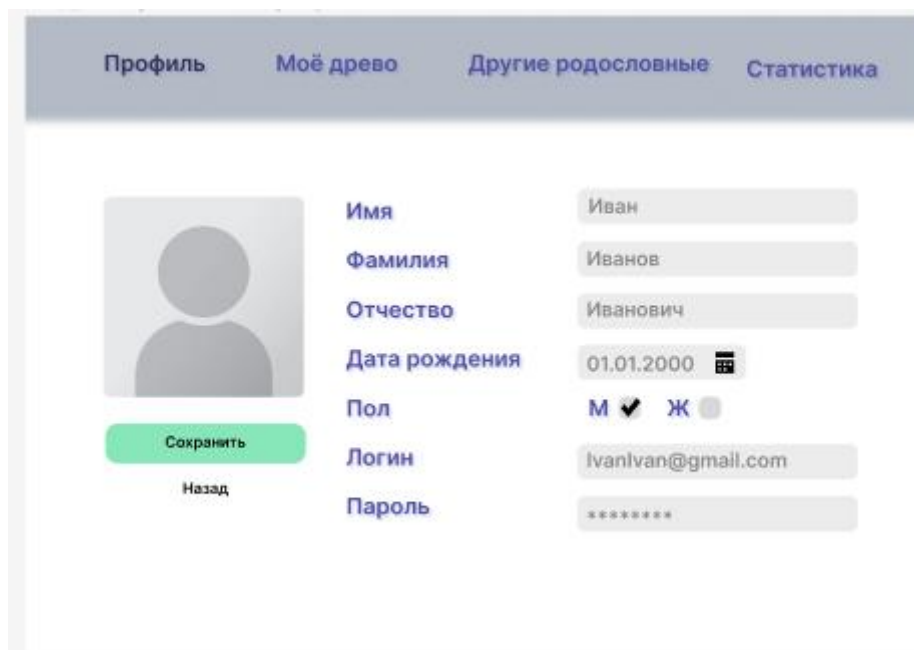
	Имя	Иван
	Фамилия	Иванов
	Отчество	Иванович
	Дата рождения	01.01.2000
	Пол	М
	Логин	IvanIvan@gmail.com
	Пароль	*****

Рисунок 1 - Экран профиля пользователя

2. Экран редактирования профиля (Рис. 2)



Профиль	Моё древо	Другие родословные	Статистика
---------	-----------	--------------------	------------




	Имя	Иван
 Назад	Фамилия	Иванов
	Отчество	Иванович
	Дата рождения	01.01.2000 
	Пол	М <input checked="" type="radio"/> Ж <input type="radio"/>
	Логин	IvanIvan@gmail.com
	Пароль	*****

Рисунок 2 - Экран редактирования профиля пользователя

3. Экран страницы дерева в табличном виде (Рис. 3)

Изображение	Информация	Родство	
	Петр Петрович Петров М 01.01.1900 - 01.01.2000	Супруга: Александра Александровна Александрова Сын: Василий Петрович Петров	
	Александра Александровна Александрова Ж 01.01.1900	Супруг: Петр Петрович Петров Сын: Василий Петрович Петров	

Рисунок 3 - Экран страницы дерева в табличном виде

4. Экран страницы дерева в виде графа (Рис. 4)

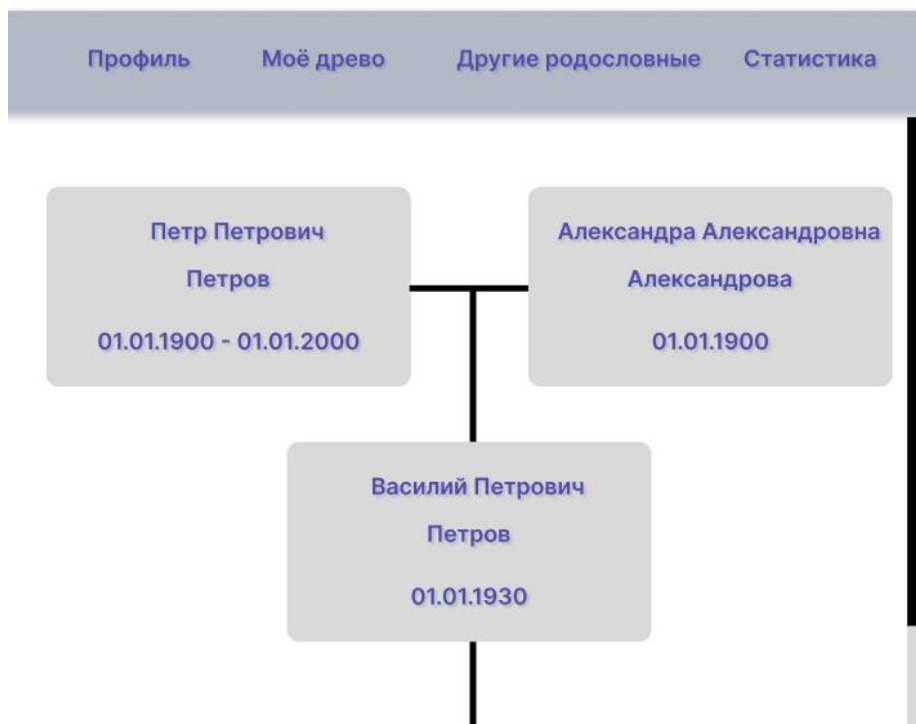


Рисунок 4 - Экран страницы дерева в виде графа

5. Экран страницы рекомендации (Рис. 5)

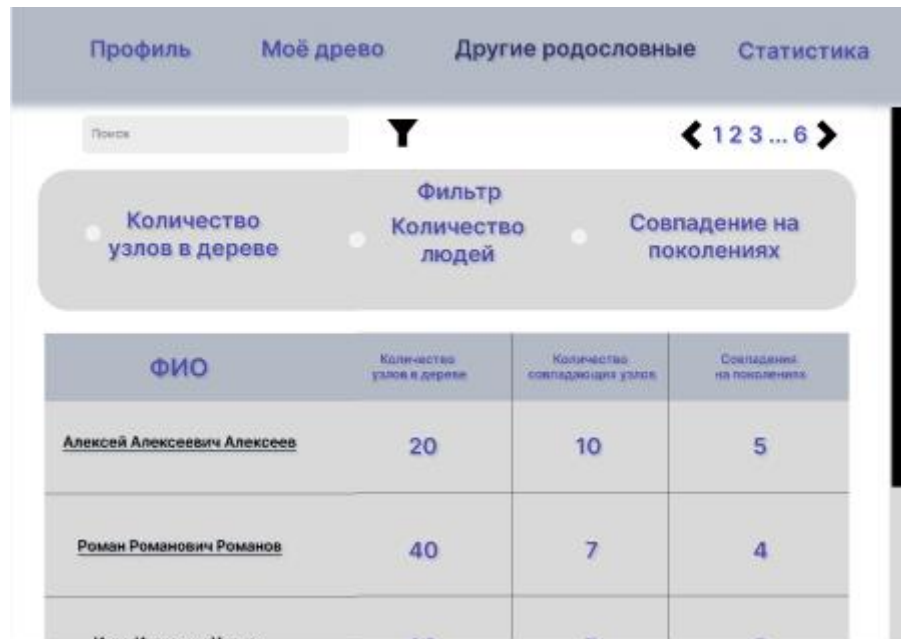


Рисунок 5 - Экран страницы рекомендаций

6. Экран страницы статистики (Рис. 6)

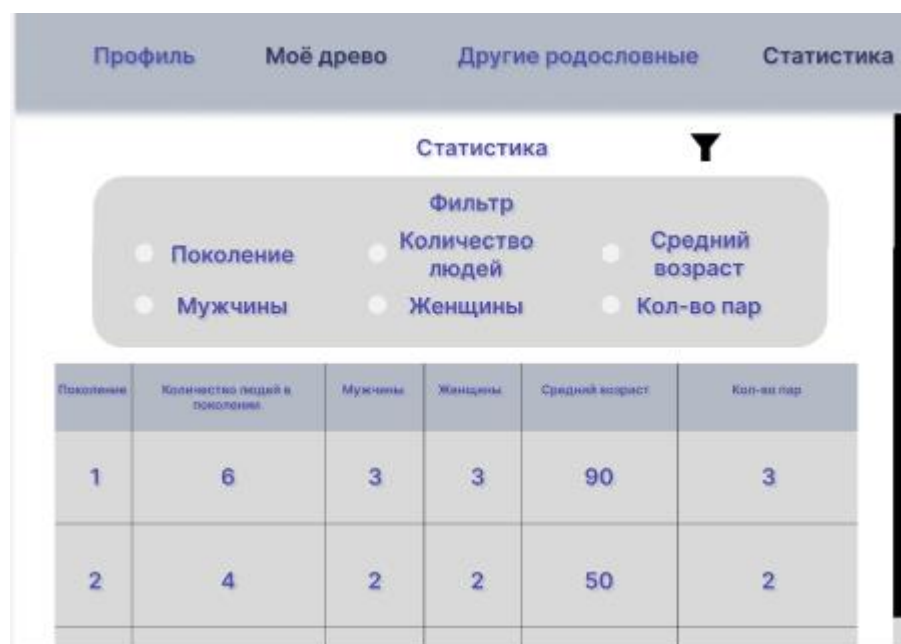


Рисунок 6 - Экран страницы статистики

7. Экран добавления и редактирования узла (Рис. 7)

Рисунок 7 - Экран добавления и редактирования узла

2.2. Сценарии использования для задачи

2.2.1. Импорт данных

2.2.1.1. Сценарий использования - “массовый импорт данных”

Действующее лицо - пользователь

Предусловие:

- Пользователь авторизовался в системе и находится на странице мое древо

Основной сценарий:

1. Пользователь видит кнопку “импорт”
2. Пользователь нажимает на иконку “импорт”
3. Появляется окно с выбором файла для импорта данных
4. Пользователь выбирает файл
5. Данные отправляются на серверную часть и успешно импортируются

Альтернативный сценарий:

1. Пользователь выбрал некорректный файл
2. Выводится сообщение о некорректном файле

2.2.1.2. Сценарий использования - “ручной импорт данных”

Действующее лицо - пользователь

Предусловие:

- Пользователь авторизовался в системе и находится на странице добавления узла

Основной сценарий:

1. Пользователь видит поле для имени, фамилии, отчества(опционально), даты рождения, даты смерти(опционально), пол, кнопку “добавить”(для добавление большего количества пар селектов) и два селекта: в первом список всех узлов дерева, во втором список всего типа родства
2. Пользователь заполняет необходимые данные и выбирает селекты
3. Пользователь нажимает на кнопку “сохранить”
4. Узел успешно добавлен

Альтернативный сценарий № 1:

1. Пользователь вводит не все необходимые данные/вводит их некорректно или не выбирает селекты
2. Выводится сообщение о некорректно введенных данных
3. Пользователь заполняет данные правильно

Альтернативный сценарий № 2:

1. Пользователь нажимает кнопку “назад”
2. Пользователь не добавляет узел

2.2.2. Представление данных

Действующее лицо - пользователь

Предусловие:

- Пользователь авторизовался в системе и находится на странице дерева

Основной сценарий:

1. Пользователь видит таблицу, в которой представлены два столбца, в первом расположено ФИО узлов, во втором расположены родство и родственная связь узла с другими.

Альтернативный сценарий :

1. Пользователь видит кнопку “Дерево”.
2. Пользователь нажимает на кнопку.
3. Пользователь переходит на страницу дерева.
4. Пользователь видит дерево, представленное в виде графа, в котором ребра подписаны степенью родства между узлами.

2.2.3. Анализ данных

Действующее лицо - пользователь

Предусловие:

- Пользователь авторизовался в системе и находится на странице дерева
- Изначально в дереве уже расположен узел пользователя

Основной сценарий:

1. Пользователь нажимает на кнопку навигации “статистика”
2. Пользователь успешно переходит на страницу статистики, на которой отображена информация о дереве по следующим критериям: поколение, количество людей в поколении, количество мужчин, количество женщин, средний возраст и количество пар.

2.2.4. Экспорт данных

Действующее лицо - пользователь

Предусловие:

- Пользователь авторизовался в системе и находится на странице “мое дерево”

Основной сценарий:

1. Пользователь видит кнопку “экспорт”
2. Пользователь нажимает на кнопку

3. Система формирует соответствующий файл в формате JSON с информацией о дереве пользователя и отправляет на клиентскую сторону.
4. Файл скачивается на устройство пользователя.

2.3. Вывод относительно нашего функционала

В результате анализа сценариев использования и самой задачи приложения был сделан вывод, что в задаче преобладают операции чтения, поскольку импорт представляет собой запись информации в базу данных, а все остальные операции направлены на чтение данной информации.

3. МОДЕЛЬ ДАННЫХ

3.1. Нереляционная модель данных - Neo4j

3.1.1. Графическое представление данных

Графическое представление нереляционной базы данных представлено на рис. 9

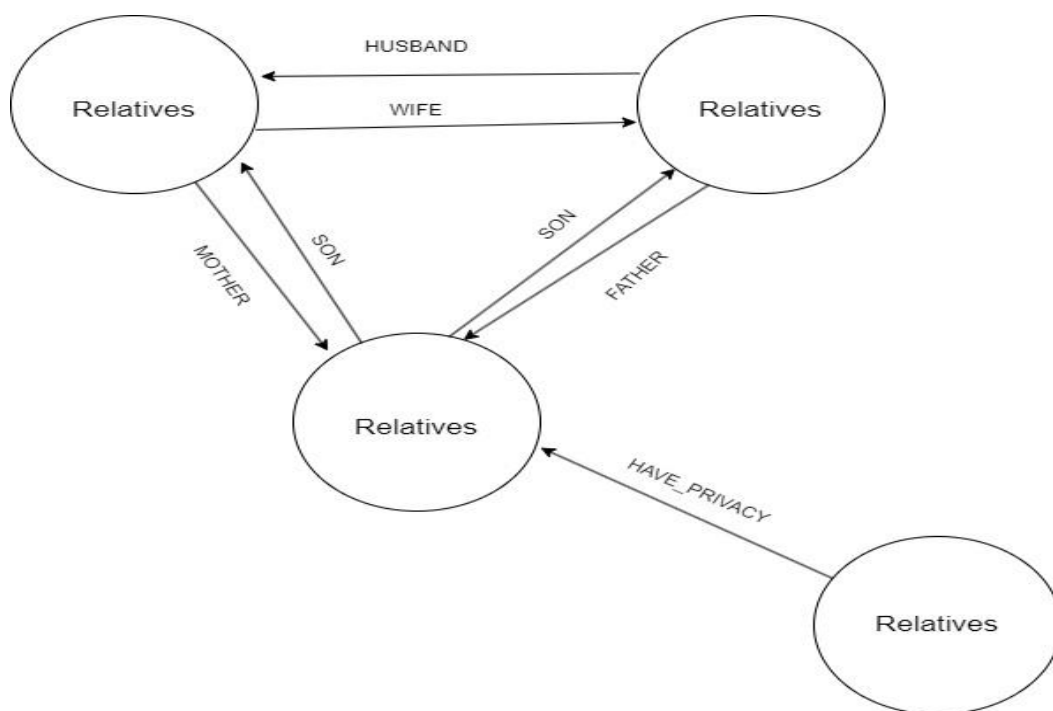


Рисунок 9 - графическое представление нереляционной базы данных

3.1.2. Описание назначений коллекций, типов данных и сущностей

Вершина: Человек, являющийся частью дерева Узел имеет следующие свойства:

- id - уникальный идентификатор узла. Тип Int. V = 4b
- UserId - идентификатор дерева. Тип Int. V = 4b
- name - имя узла. Тип String. V = 128b
- surname - фамилия узла. Тип String. V = 128b
- patronymic - отчество узла. Тип String. V = 128b
- dateOfBirth - дата рождения пользователя. Тип Date. V = 3b

- dateOfDeath - дата смерти узла. Тип Date. V = 3b
- gender - пол пользователя. Тип String. V = 2b
- generation - номер поколения. Тип Int. V = 4b
- login - логин пользователя. Тип String. V = 64b
- password - пароль пользователя. Тип String. V = 64b

Связь между людьми, является ребром, соединяющим 2 узла.

- id - уникальный идентификатор узла. Тип Int. V = 4b

От каждого узла могут выходить рёбра, имеющие следующие метки:

- SON - узел является сыном относительно узла, в которого входит ребро
- DAUGHTER - узел является дочерью относительно узла, в которого входит ребро
- FATHER - узел является отцом относительно узла, в которого входит ребро
- MOTHER - узел является матерью относительно узла, в которого входит ребро
- WIFE - узел является женой относительно узла, в которого входит ребро
- HUSBAND - узел является мужем относительно узла, в которого входит ребро
- BROTHER - узел является братом относительно узла, в которого входит ребро
- SISTER - узел является сестрой относительно узла, в которого входит ребро
- HAVE_PRIVACY - узел имеет доступ к дереву узла, в которого входит ребро

3.1.3. Оценка удельного объема информации, хранимой в модели

Для создания одного пользователя потребуется, в среднем:

- Один узел “Relative” = 526b

Для создания одного родственника(узла) потребуется, в среднем:

- Один узел “Relative” = 404b (без полей логина, пароля, userId и privacy)
- Два ребра = 8b

Средний объем для Пользователя $V = 526b$. Средний объем связи между узлами $V = 412b$.

объем данных для хранения N_n пользователей и N_w узлов:

- $V(N_n, N_w) = V_n * N_n + V_w * N_w$

Объем данных для хранения Пользователей с узлами:

Пусть системой пользуется x пользователей и у каждого дерево заполнено на 10 человек, а также предоставлен доступ 10 другим пользователям, тогда получаем:

- $V(x) = 526 * x + 412 * 10 * x + 4 * 10 * x = 4686 * x$

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

Запрос на добавление пользователя:

```
CREATE (user:Relative {name: "Иван", surname: "Иванов", patronymic: "Иванович", dateOfBirth: date("2002-08-12"), login: "qwerty1287", password: "sfwf3rf13!", gender: "M", generation: 1})
```

Запрос на добавление узла и связи с ним:

```
MATCH (first:Relative) WHERE ID(first) = id1  
CREATE (second:Relative {name: "Петр", surname:
```



```
"Петров",patronymic: "Петрович",dateOfBirth: date("1978-07-16"),dateOfDeath: date("2020-01-01"), gender: "М",generation: 2})) CREATE(second)-[w:FATHER]->(first) CREATE(first)-[y:SON]->(second) RETURN y
```

Запрос для поиска количество узлов, имеющих связь с пользователем(узлов дерева):

```
MATCH(m:Relatives) WHERE m.UserID = UserId1 RETURN COUNT(m)
```

Запрос на количество совпадающих узлов между двумя деревьями по полям(имя, фамилия, отчество, год рождения и пол) с учётом открытого доступа дерева:

```
MATCH(m:Relatives), (n:Relatives) WHERE m.UserID = UserId1 AND n.UserID = UserId2 AND n.name = m.name AND n.patronymic = m.patronymic AND n.surname = m.surname AND n.datebirth = m.datebirth AND n.gender = m.gender RETURN COUNT(n)
```

3.2. Реляционная модель данных

3.2.1. Графическое представление данных

Графическое представление реляционной базы данных представлено на рис. 10

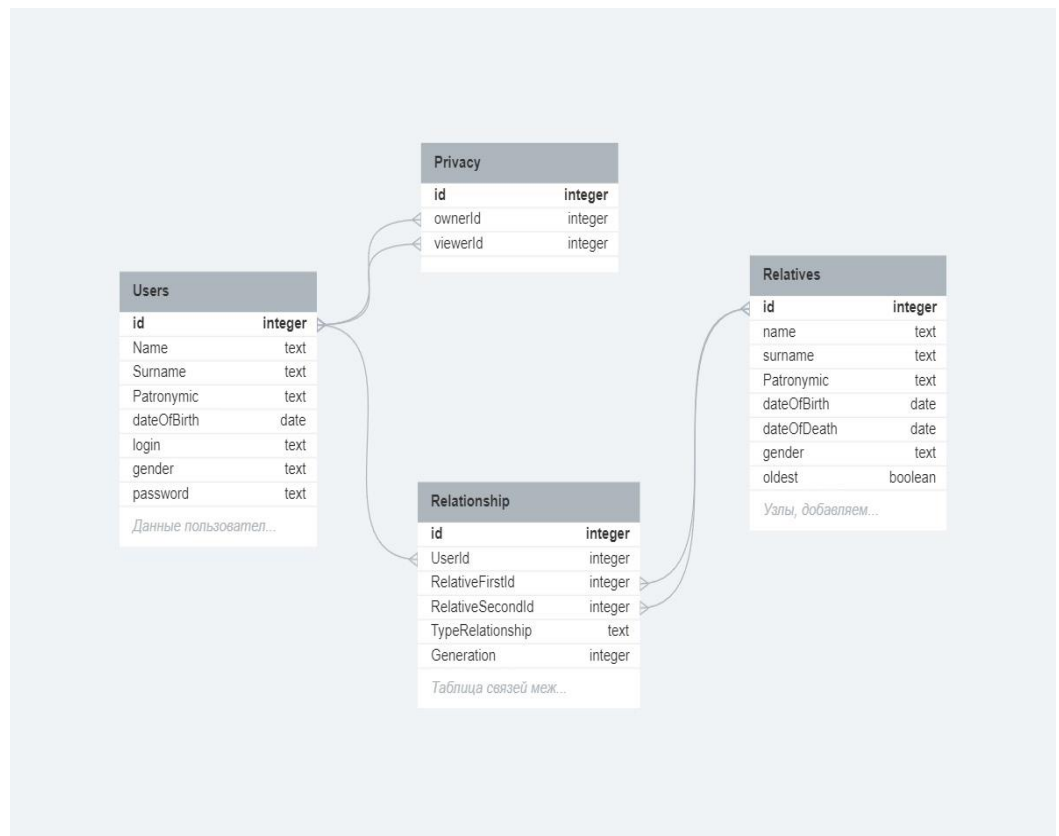


Рисунок 10 - графическое представление реляционной базы данных

3.2.2. Описание назначений коллекций, типов данных и сущностей

С учётом того, что кириллица занимает 2 байта на символ

Таблица "Users"

- id - уникальный идентификатор пользователя. Тип Int. V = 4b
- name - имя пользователя. Тип nvarchar(64). V = 128b
- surname - фамилия пользователя. Тип nvarchar(64). V = 128b
- patronymic - отчество пользователя. Тип nvarchar(64). V = 128b
- dateOfBirth - дата рождения пользователя. Тип Date. V = 3b
- login - логин пользователя. Тип varchar(64). V = 64b
- password - пароль пользователя. Тип varchar(64). V = 64b
- gender - пол пользователя. Тип nvarchar(1). V = 2b

Таблица "Privacy"

- id- уникальный идентификатор доступа. Тип Int. V = 4b
- ownerId- уникальный идентификатор пользователя от кого доступ. Тип Int. V = 4b
- viewerId- уникальный идентификатор пользователя к кому есть доступ. Тип Int. V = 4b

Таблица "Relatives"

- id - уникальный идентификатор узла. Тип Int. V = 4b
- name - имя узла. Тип nvarchar(64). V = 128b
- surname - фамилия узла. Тип nvarchar(64). V = 128b
- patronymic - отчество узла. Тип nvarchar(64). V = 128b
- dateOfBirth - дата рождения пользователя. Тип Date. V = 3b
- dateOfDeath - дата смерти узла. Тип Date. V = 3b
- gender - пол пользователя. Тип nvarchar(1). V = 2b
- oldest - атрибут самого старшего поколения. Тип Boolean. V=1b

Таблица "Relationship"

- id - уникальный идентификатор отношений. Тип Int. V = 4b
- UserId - публичный ключ пользователя. Тип Int. V = 4b
- RelativeFirstId - публичный ключ первого узла. Тип Int. V = 4b
- RelativeSecondId - публичный ключ второго узла. Тип Int. V = 4b
- TypeRelationship - тип родства. Тип nvarchar(64). V = 128b
- generation - поколение. Тип int. V = 4b

3.2.3. Оценка удельного объема информации, хранимой в модели

Для создания одного пользователя потребуется, в среднем:

- одна запись в таблице "users" = 521b

Для создания одного родственника(узла) потребуется, в среднем:

- одна запись в таблице "relatives" = 397b
- одна запись в таблице "relationship" = 148b

Средний объём для Пользователя $V = 521b$. Средний объем связи между узлами $V = 693b$ (2 записи для одной связи). Средний объем для предоставления данных о дереве пользователем $V = 12b$.

Объем данных для хранения N_n пользователей и N_w узлов

- $V(N_n, N_w) = V_n * N_n + V_w * N_w$

Пусть системой пользуется x пользователей и у каждого дерево заполнено на 10 человек, а также доступ предоставлен 10 пользователям тогда получаем:

- $V(x) = 521 * x + 693 * 10 * x + 12 * 10 * x = 7571 * x$

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

Запрос на добавление пользователя:

```
INSERT Users(id,name, surname,
patronymic,gender,dateOfBirth,login,password) VALUES
(id1,name1, surname1,
patronymic1,gender1,dateOfBirth1,login1,password1);
```

Запрос на добавление узла и связи с ним:

```
INSERT Relatives(id,name, surname,
patronymic,dateOfBirth,dateOfDeath,gender,oldest) VALUES
(id1,name1, surname1,
patronymic1,dateOfBirth1,dateOfDeath1,gender1,oldest1);
INSERT
Relationship(id,UserId,RelativeFirstId,RelativeSecondId,
TypeRelationship,generation)
VALUES(id1,UserId1,RelativeFirstId1,RelativeSecondId1,Ty
peRelationship1,generation1);
```

Запрос для поиска количество узлов, имеющих связь с пользователем(узлов дерева):

```
SELECT COUNT(*) FROM Relationship WHERE UserId =  
UserId1;
```

Запрос на количество совпадающих узлов между двумя деревьями по полям(имя, фамилия, отчество, год рождения и пол) с учётом открытого доступа дерева:

```
SELECT viewerId FROM Privacy WHERE ownerId = u1.id  
UNION SELECT COUNT(*) AS common_relatives_count FROM (  
SELECT u1.id AS user1_id, u2.id AS user2_id FROM Users  
AS u1, Users AS u2 WHERE u1.id = :user1_id AND u2.id =  
:user2_id ) AS users JOIN ( SELECT rel1.id AS  
common_relative_id FROM Relatives AS rel1 JOIN  
Relationship AS r1 ON rel1.id = r1.RelativeFirstId JOIN  
Users AS u1 ON r1.UserId = users.user1_id WHERE u2.id IN  
viewerId AND EXISTS ( SELECT 1 FROM Relatives AS rel2  
JOIN Relationship AS r2 ON rel2.id = r2.RelativeFirstId  
JOIN Users AS u2 ON r2.UserId = users.user2_id WHERE  
rel1.name = rel2.name AND rel1.surname = rel2.surname  
AND rel1.patronymic = rel2.patronymic AND  
rel1.dateOfBirth = rel2.dateOfBirth AND rel1.gender =  
rel2.gender ) ) AS common_relatives;
```

3.3. Сравнение моделей

3.3.1. Удельный объем информации

Neo4j требует заметно меньше памяти - как указано в примере для x пользователей у которых по 10 родственников и доступ предоставлен 10 другим пользователям в SQL занимает $7571 * x$, а в Neo4j $4686 * x$. Память уменьшилась на 38%.

3.3.2. Запросы по отдельным юзкейсам

Neo4j также выигрывает по качеству запросов: Не требует объединения таблиц для получения информации о пользователях и узлах.

3.3.3. Вывод

Neo4j подходит для данной задачи лучше чем SQL, так как замечен очевидный выигрыш по памяти и по составлению запросов для БД.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание

Back-end представляет из себя приложение, разработанное на языке программирования JavaScript и в фреймворке Express. Приложение основано на Model — View — Controller архитектуре. MVC – это шаблон проектирования, который построен на основании принципа сохранения представления данных. Согласно этому принципу, данные хранятся отдельно от методов, взаимодействующих с этими данными. За основную базу данных взята Neo4j, настроено взаимодействие между JavaScript и базой данных. Сервис позволяет создавать генеалогические деревья, добавлять в них узлы, редактировать их, создавать между ними связи, а также просматривать деревья других пользователей для поиска общих родственников. Также на странице статистики можно смотреть информацию о нашем дереве, сколько поколений в дереве, количество мужчин и женщин в определенном поколении, а также количество пар в поколениях.

Front-end основан на фреймворке Vue.js с дополнительным использованием UI библиотеки компонентов Vuetify. Также front-end основан на Server-Side Rendering. Мы предоставляем пользователю уже готовую страницу со всеми стилями и данными, когда он делает запрос внутри нашего приложения.

4.2. Используемые технологии


БД: Neo4j

Back-end: JavaScript, Express, neo4j-driver

Front-end: JavaScript, CSS, Vue.js, Vuetify

4.3. Снимки экрана приложения

ПРОФИЛЬ МОЁ ДЕРЕВО ДРУГИЕ РОДОСЛОВНЫЕ СТАТИСТИКА



Имя

Фамилия

Отчество

Дата рождения

Дата смерти

Пол ☐ М ☐ Ж

Родство

Рисунок 14 – Страница добавления узла

ПРОФИЛЬ МОЁ ДЕРЕВО ДРУГИЕ РОДОСЛОВНЫЕ СТАТИСТИКА

Search

ФИО	Количество узлов в дереве	Количество совпадающих узлов	Совпадения на поколениях
Невс Греков Кроносович	5	0	0
Ревс Греков Кроносович	5	1	1
Зевс Греков Кроносович	7	4	3

Items per page: 10 1-3 of 3 < < > >

Рисунок 15 – Страница других деревьев

ПРОФИЛЬ МОЁ ДЕРЕВО ДРУГИЕ РОДОСЛОВНЫЕ СТАТИСТИКА

Статистика

Поколение	Количество людей в поколении	Мужчины	Женщины	Средний возраст	Количество пар
0	2	1	1	601	1
1	2	1	1	517	1
2	1	1	0	323	0

Items per page: 5 1-3 of 3 < < > >

Рисунок 16 – Страница статистики дерева

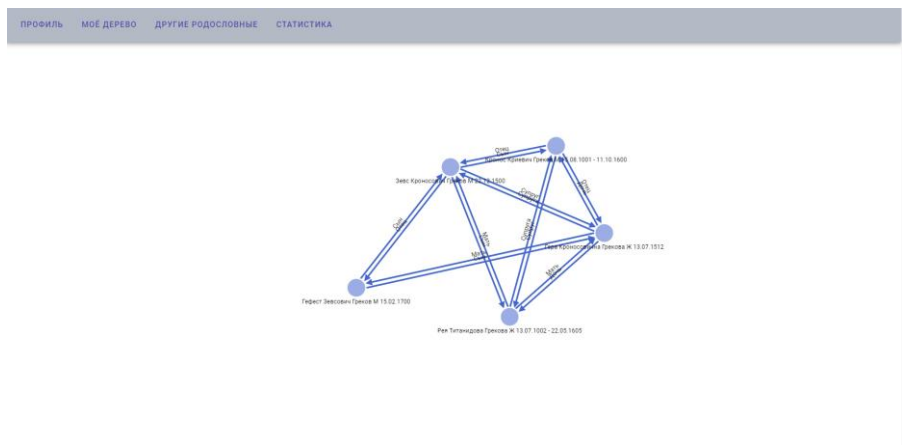


Рисунок 17 – Страница дерева представленного в виде графа

Рисунок 18 – Страница редактирования узла дерева

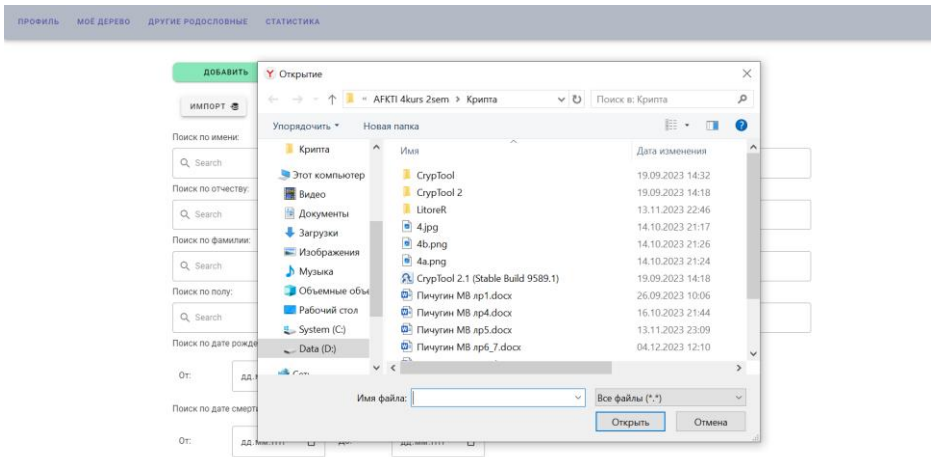


Рисунок 19 – Окно импорта дерева

Имя	Фамилия	Отчество	Пол	Дата рождения	Дата смерти	Вспомогательная информация	Действия
Гегест	Зевсович	Греков	М	15.02.1700		Кроносина Грекова	
Кронос	Кривич	Греков	М	15.08.1001	11.10.1600	["Отец: Зевс Кроносич Греков", "Мать: Гера Кроносичина Грекова"]	✕
Рей	Титандова	Грекова	Ж	13.07.1002	22.05.1605	["Сын: Зевс Кроносич Греков", "Дочь: Гера Кроносичина Грекова", "Супруга: Рей Титандова Грекова"]	✕
Гера	Кроносичина	Грекова	Ж	13.07.1512		["Сын: Зевс Кроносич Греков", "Супруг: Кронос Кривич Греков", "Дочь: Гера Кроносичина Грекова"]	✕
						["Супруг: Зевс Кроносич Греков", "Отец: Кронос Кривич Греков", "Мать: Рей Титандова Грекова", "Сын: Гегест Зевсович Греков"]	✕

Рисунок 20 – Окно удаления узла дерева


ПРОФИЛЬ	МОЁ ДЕРЕВО	ДРУГИЕ РОДОСЛОВНЫЕ	СТАТИСТИКА
<div>  <div> <div>Имя</div> <div>Зевс</div> </div> <div> <div>Фамилия</div> <div>Греков</div> </div> <div> <div>Отчество</div> <div>Кроносович</div> </div> <div> <div>Дата рождения</div> <div>1500-12-22</div> </div> <div> <div>Пол</div> <div>М</div> </div> <div> <div>Логин</div> <div>jevisCoo0228</div> </div> <div> <div>Пароль</div> <div>*****</div> </div> </div>			
<div>РЕДАКТРИРОВАТЬ</div>			

Рисунок 21 – Страница профиля


ПРОФИЛЬ	МОЁ ДЕРЕВО	ДРУГИЕ РОДОСЛОВНЫЕ	СТАТИСТИКА
<div>  <div> <div>Имя</div> <div>Зевс</div> </div> <div> <div>Фамилия</div> <div>Греков</div> </div> <div> <div>Отчество</div> <div>Кроносович</div> </div> <div> <div>Дата рождения</div> <div>22.12.1500</div> <div></div> </div> <div> <div>Пол</div> <div> <input checked="" type="radio"/> М <input type="radio"/> Ж </div> </div> <div> <div>Логин</div> <div>jevisCoo0228</div> </div> <div> <div>Пароль</div> <div> ***** <div></div> </div> </div> </div>			
<div>СОХРАНИТЬ</div>			
<div>НАЗАД</div>			

Рисунок 22 – Страница редактирования дерева

5. ВЫВОДЫ

5.1. Достигнутые результаты

В ходе работы было разработано приложение веб-сервиса генеалогических деревьев с использованием нереляционной базы данных Neo4j. Была реализована вся заявленная функциональность: корректная работа с Neo4j, управление деревом, их элементами и другими деревьями. Добавлена возможность локального разворота приложения с помощью docker-compose.

5.2. Недостатки и пути для улучшения полученного решения

Представление дерева в виде графа не располагает узлы деревьев в читаемом виде, их нужно перетаскивать, чтобы получить полное представление о дереве. Эту проблему можно решить путем более глубокого изучения используемой библиотеки v-network-graph.

Наличие повторяющихся кусков кода как внутри back-end, так и внутри front-end. Огромный недостаток состоит в том, что при внесении каких-то изменений в повторяющиеся куски – необходимо много раз копировать изменения и вставлять в другие файлы, что увеличивает объём работы в разы. Путь для улучшения – вынести эти куски кода в отдельные файлы и далее их переиспользовать их через подключение.

5.3. Будущее развитие решения

Улучшение front-end вёрстки, поскольку большей части страниц необходим обновлённый дизайн и более качественный пользовательский интерфейс. Вынесение часто повторяющихся кусков кода в отдельные файлы, чтобы переиспользовать файлы и ускорить внесение изменений в код. Более глубокое изучение библиотеки визуализации графов для автоматизированного расположение узлов в древовидном виде.

6. ПРИЛОЖЕНИЯ

6.1. Документация по сборке и развертыванию приложения

1. Склонировать репозиторий (указан в списке литературы)
2. Зайти в папку `./nosql2h23-genealogy/`
3. Установить `docker` и `docker-compose` с помощью следующих

команд:

- `sudo apt update`
- `sudo apt install docker-ce -y`
- `sudo systemctl status docker`
- `sudo apt-get install docker-compose`

4. Запустить локальный разворот приложения через `docker-compose`

с помощью следующих команд:

- `sudo docker-compose build --no-cache`
- `sudo docker-compose up -d`

5. Перейти в любом удобном браузере по адресу: `127.0.0.1`

6.2. Инструкция для пользователя

1. Используя свой логин и пароль войти в систему на странице авторизации ИЛИ зарегистрироваться на странице регистрации, после чего авторизоваться.

2. Перейти на страницу «Дерево»
3. Нажать кнопку добавить для добавления узла
4. Начать построение дерева
5. Повтор пунктов 2-4

7. ЛИТЕРАТУРА

1. Ссылка на github проекта - <https://github.com/moevm/nosql2h23-genealogy>
2. Документация Neo4j - <https://neo4j.com/docs/>
3. Документация Vue.js - <https://ru.vuejs.org/v2/guide/>
4. Документация JavaScript - <https://devdocs.io/javascript/>
5. Документация Express - <https://expressjs.com/en/api.html>
6. Документация Docker - <https://docs.docker.com/>