**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: ИС построения генеалогических деревьев (Mongo)**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 7382 |  | Государкин Я.С. |
|  |  | Петрова А.С. |
|  |  | Токарев А.П. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2020

**ЗАДАНИЕ**

**на** ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

|  |  |  |
| --- | --- | --- |
| Студенты Государкин Я.С., Петрова А., Токарев А.П. | | |
| Группа 7382 | | |
| Тема работы: ИС построения генеалогических деревьев (Mongo) | | |
| Исходные данные:  Реализовать информационную систему построения генеалогических деревьев с визуализацией по данным из БД | | |
| Содержание пояснительной записки:  «Содержание», «Введение», «Качественные требования к решению», «Сценарий использования», «Модель данных», «Разработка приложения», «Заключение», «Список использованных источников» | | |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц. | | |
| Дата выдачи задания: 17.09.2020 | | |
| Дата сдачи реферата: 27.12.2020 | | |
| Дата защиты реферата: 27.12.2020 | | |
| Студенты гр. 7382 |  | Государкин Я.С. |
|  |  | Петрова А.С. |
|  |  | Токарев А.П. |
| Преподаватель |  | Заславский М.М. |

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 4 |
| 1. | Качественные требования к решению | 5 |
| 2. | Сценарии использования | 0 |
| 2.1. | Макет пользовательского интерфейса | 0 |
| 2.2. | Описание возможных сценариев использования | 0 |
| 3. | Модель данных | 0 |
| 3.1. | Нереляционная модель данных | 0 |
| 3.2.  4. | Аналог модели данных для реляционной БД  Разработанное приложение | 0 |
|  | Заключение | 0 |
|  | Список использованных источников | 0 |

**введение**

Реализовать веб-приложение с возможность построения генеалогических деревьев, с получением дерева для каждой иерархии и для отдельного человека, а также с возможностью получения статистических данных.

Backend приложения написан на языке Python3 c использованием СУБД Mongo.

Frontend – JS, с использованием React и Nginx.

1. **КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ**

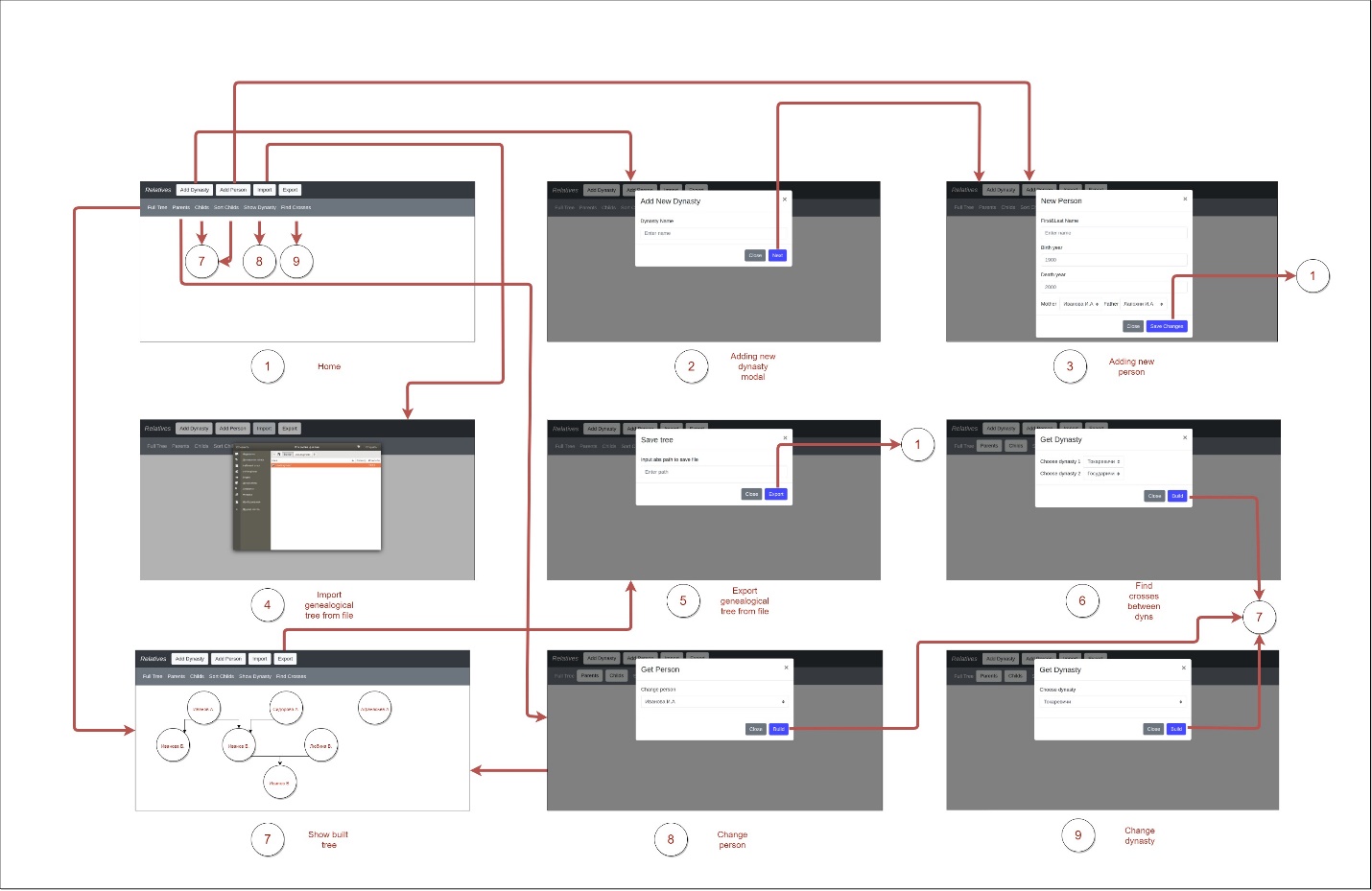
Веб-приложение с некоторым UI, который позволяет:

* + добавлять деревья
  + поиск пересечений между династиями
  + поиск пересечений для определенного человека
  + сортировка выбранного дерева дерева

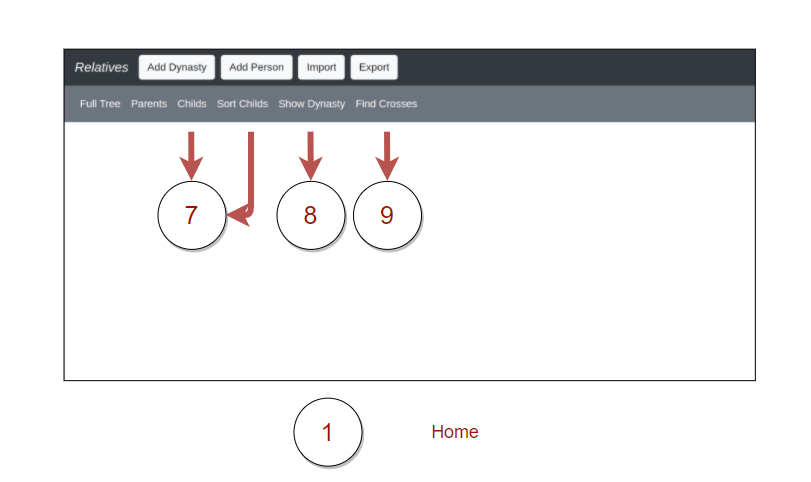
Использование Mongo DB в качестве СУБД.

1. **СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ**

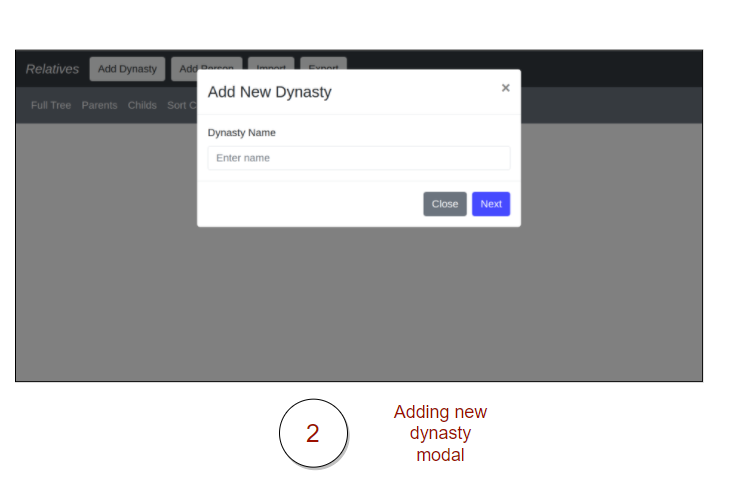
**2.1. Макет пользовательского интерфейса**



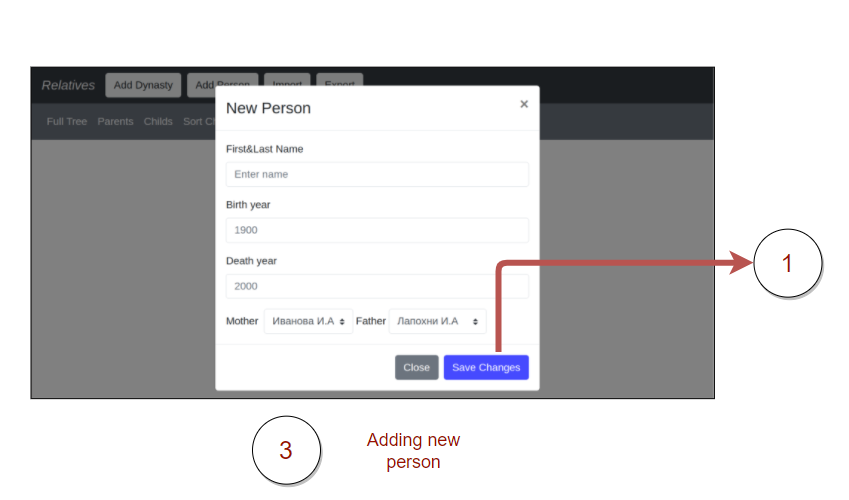
1.Домашняя страница:



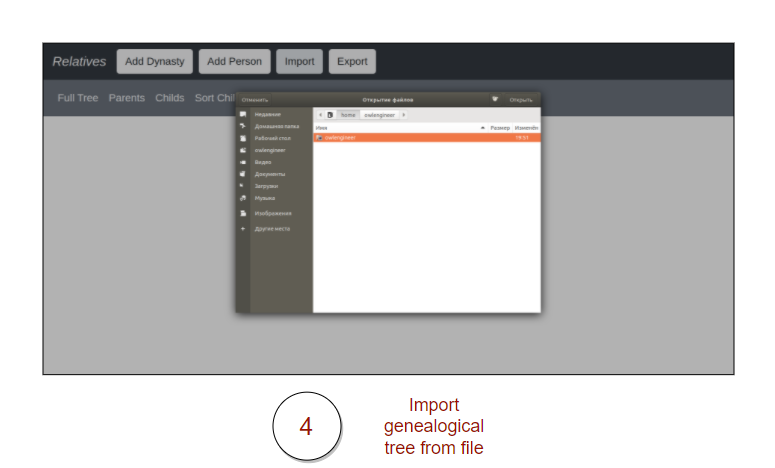
2. Добавление новой династии:



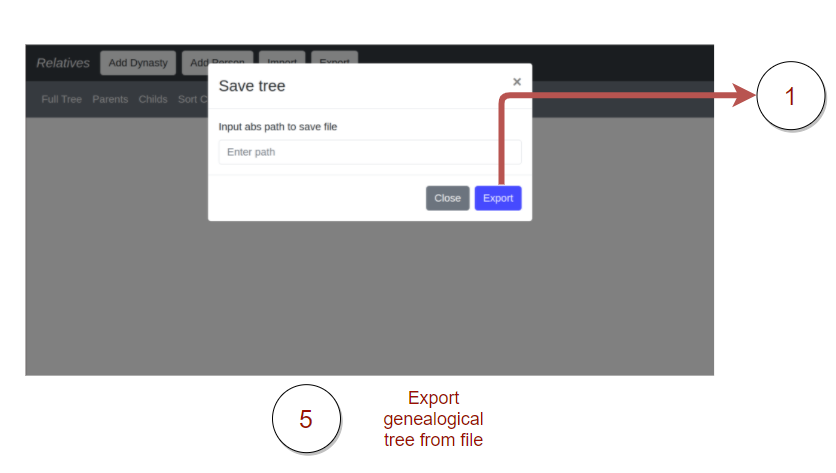
3. Добавление нового человека:



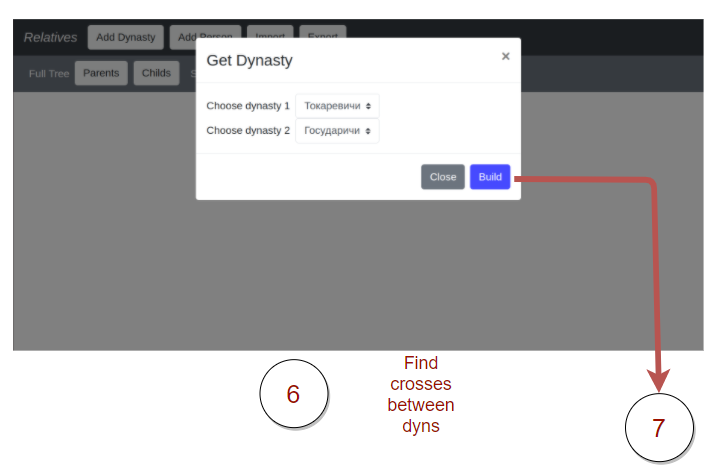
4. Импортировать дерево из файла:



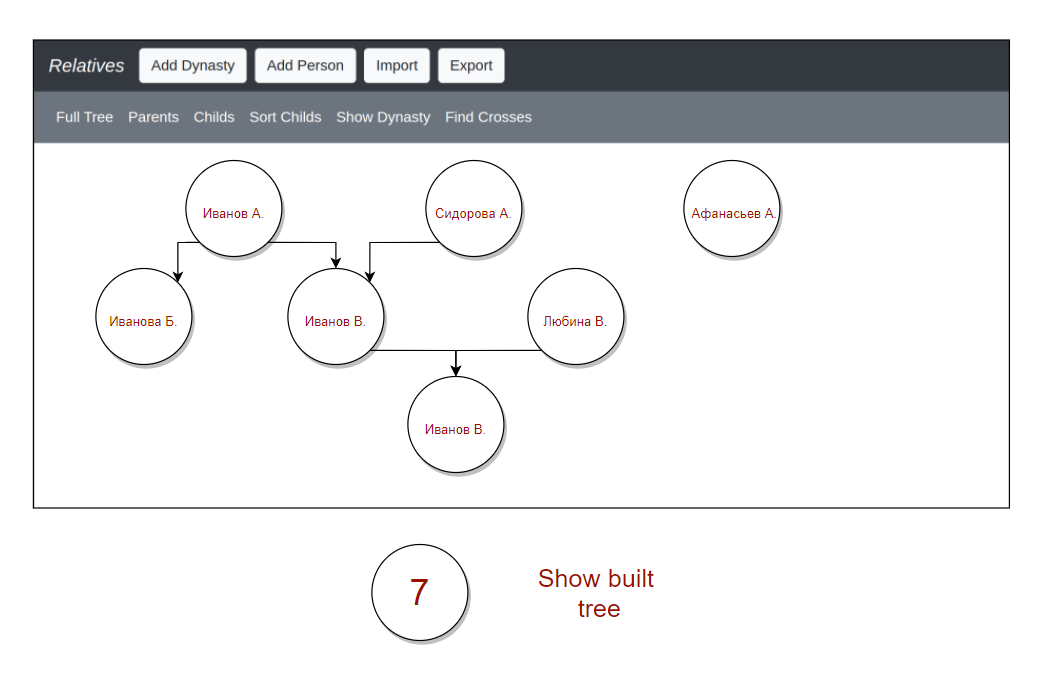
5. Сохранить генеалогическое дерево в файл:



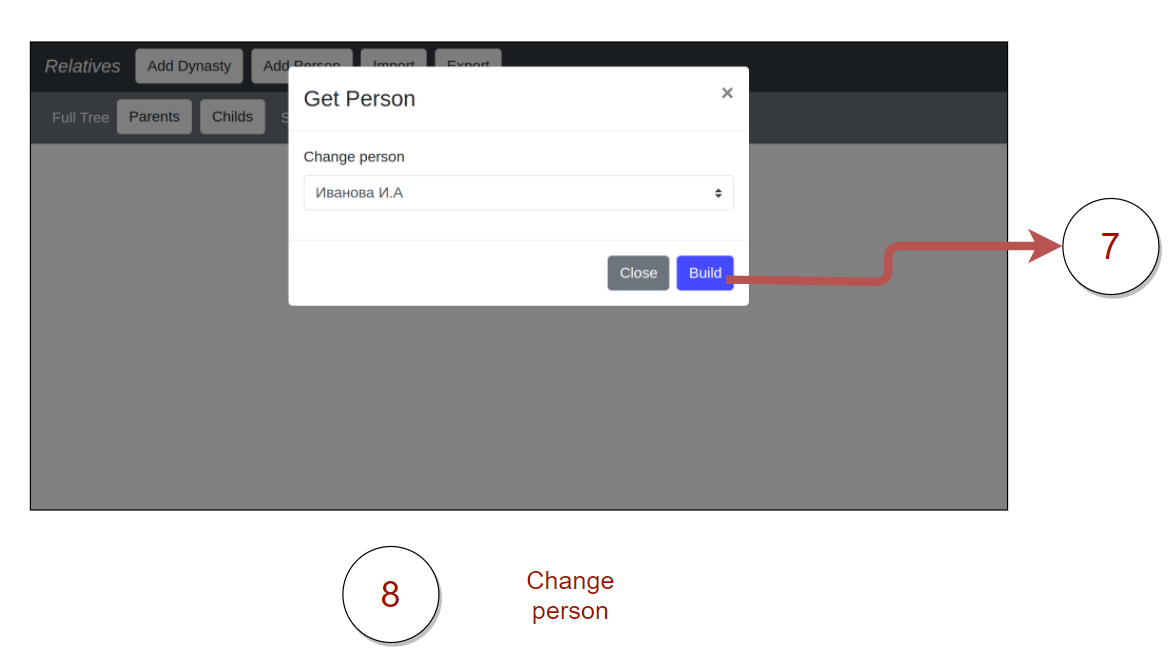
6. Найти пересечения между династиями:



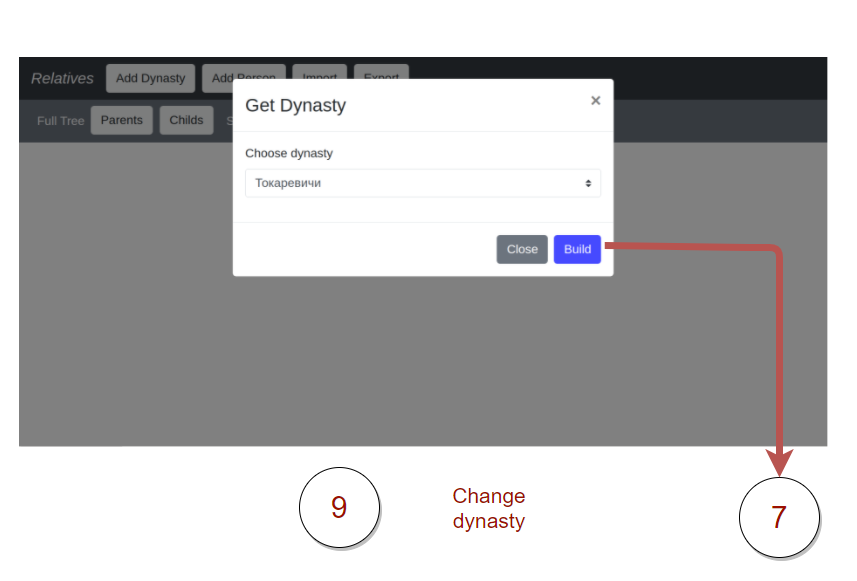
7. Просмотреть построенное дерево:



8. Выбрать человека из списка:



9. Выбрать династию из списка:

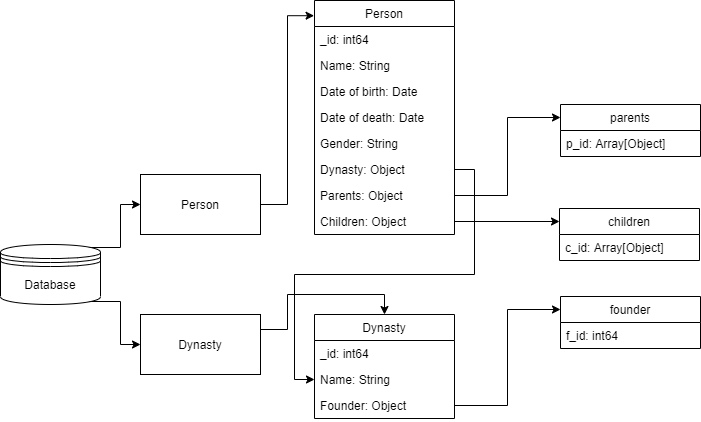


**2.2 Описание возможных сценариев использования.**

1. Добавление новой династии (Название, картинка). Когда создается династия - обязательно создается человек-основатель. Далее важно - если основатель - мужчина, то династия наследуется по мужской линии, если женщина, то по женской.
2. Добавление нового человека в базу (ФИО, год рождения и смерти (если умер)). Можно указать родителей (мама и папа, тут не будем учитывать усыновления и различные отклонения от физиологической нормы), но необязательно. Автоматически определяется принадлежность человека к династии на основе правила выше. Если ситуация спорная (оба родителя имеют разные династии), то ребенок получает династию с наибольшей численностью людей.
3. Отображение на графе для конкретного человека:
   1. Дерева от детей и ниже
   2. Дерево предков
   3. Всего дерева целиком
4. Каждое из деревьев можно показать в режимах:
   1. без сортировки
   2. сортировка детей по старшинству слева направо (возраст определяется как разница текущей даты и даты рождения)
5. Отображение на графе:
   1. Всю династию (для каждой династии отдельно)
   2. Две любые династии и поиск пересечений (общих детей)
6. **МОДЕЛЬ ДАННЫХ**

**3.1 Нереляционная модель данных**

**Mongo DB**



Модель состоит из 2 коллекций:

Person - хранит информацию о человеке

* \_id - int64 - уникальный идентификатор человека
* Name - String - имя
* Date of birth - Date - дата рождения
* Date of death - Date - дата смерти
* Gender - String - пол (мужской/женский)
* Dynasty - Object - название династии, к которой принадлежит человек
* Parents - Object - родители, если есть
  + - p\_id - Array[Object] - список родителей
* Children - Object - дети, если есть
  + - с\_id - Array[Object] - список детей
* Dynasty - хранит информацию о династии
* \_id - int64 - уникальный идентификатор династии
* Name - String - название династии
* Founder - Object - основатель династии
  + - f\_id - int64 - идентификатор основателя

**Оценка объема:**

1) PERSON

* \_id - int64 V=8b
* Name - String - V=50b (в среднем случае)
* Date of birth - Date - 32b
* Date of death - Date - 32b
* Gender - String - пол (m/f) - V=1b
* Dynasty - Object - 12b
* Parents - Object - родители, если есть - 12b \* 0/1/2
* Children - Object - дети, если есть - 12b \* количество детей

2) DYNASTY

* \_id - int64 - V=8b
* Name - String - V=50b
* Founder - Object V=12b

В среднем будем считать, что у каждого человека есть два ребенка, количество человек – *N*.

Допустим, что в нашей БД династий в 5 раз меньше, чем людей, тогда можно посчитать примерный объем.



**Примеры запросов:**

*Добавить человека*

db.person.insert\_many([{

'\_id' : id,

'Name' : Name,

'Date of birth' : 12.12.1200,

'Date of death' : 24.05.1245,

'Gender' : m,

'Dynasty' : Dynasty,

'Parents' : parents,

'Children' : children

}])

*Добавить династию*

db.dynasty.insert\_many([{

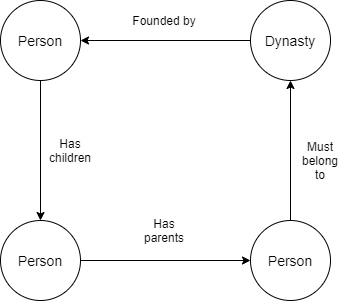
'\_id' : id,

'Name' : Dynasty,

'Founder' : Name

}])

**Neo4jl**



Модель состоит из двух сущностей - Person и Dynasty. Они связаны между собой ссылками. Каждый Person должен принадлежать к Dynasty, а Dynasty должна иметь основателя. Метками узлов являются:

1. Person:

* \_id - short - уникальный идентификатор человека
* Name - String - имя
* Date of birth - String - дата рождения
* Date of death - String - дата смерти
* Gender - Char - пол (m/f)

1. Dynasty:

* \_id - short - уникальный идентификатор династии
* Name - String - название династии

**Оценка объема:**



**Примеры запросов:**

*Добавить человека:*

CREATE (person{...})

*Создать отношение:*

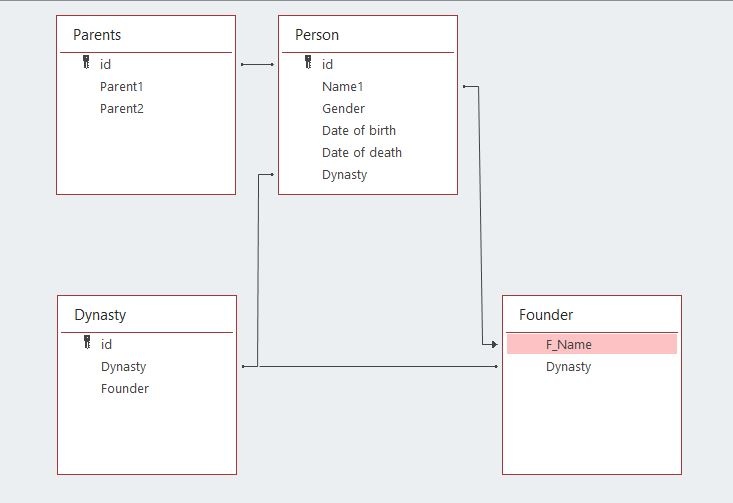
MATCH (a:person), (b:dynasty)

WHERE a.name = "Person" AND b.name = "Dynasty"

CREATE (a)-[:Belongs\_to]->(b)

RETURN a,b

# 3.2 Аналог модели данных для реляционной БД.



В такой базе данных придется хранить несколько таблиц - Person, Psrents, Dynasty, Founder. Получить список детей можно с помощью запроса между таблицами Person и Parents. Создать для этого отдельную таблицу не получится, так как у человека может либо ни одного ребенка, либо 10, и предсказать это сложно.

1. Person

* id - уникальный идентификатор
* Name1 - имя человека (короткий текст)
* Gender - пол (короткий текст)
* Date of birth - дата рождения (дата)
* Date of death - дата смерти (дата)
* Dynasty - династия, к которой принадлежит человек (короткий текст)

1. Dynasty

* id - уникальный идентификатор династии
* Name - название династии (короткий текст)
* Founder - основатель династии (короткий текст)

1. Founder

* F\_Name - имя основателя (короткий текст)
* Dynasty - (короткий текст) - название династии

1. Parents

* id - уникальный идентификатор человека
* parent1 - имя первого родителя
* parent2 - имя второго родителя

**Оценка объема:**

V = 4N + 50N + N + 82N + 50N + 4N/5 + 50N/5 +50N/5 + 50N/5 + 50N/5 + 4N +100N = 244,8N

N = 16

V = 3916,8b

V(фактический)= 417792b

Избыточность = 417792/3916,8 = 106,6

**Примеры запросов:**

Удалить из таблицы человека с id равным 10:

DELETE FROM Person WHERE \_id = '12'

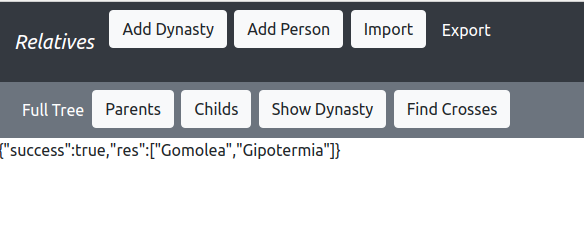
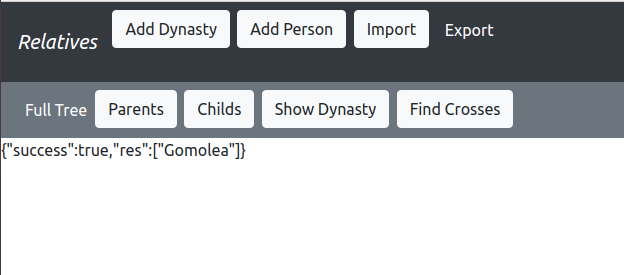
**Оценка:**

Для задачи построения генеалогических деревьев БД на SQL не подходит, так как она не дает возможности построения графов, а так же очень сильно проигрывает по памяти.

При сравнении Mongo DB и Neo4jl можно сказать, что по памяти и по количеству запросов себя показывает лучше Neo4jl, т.к. взаимодействие между сущностями в данной СУБД гораздо проще.

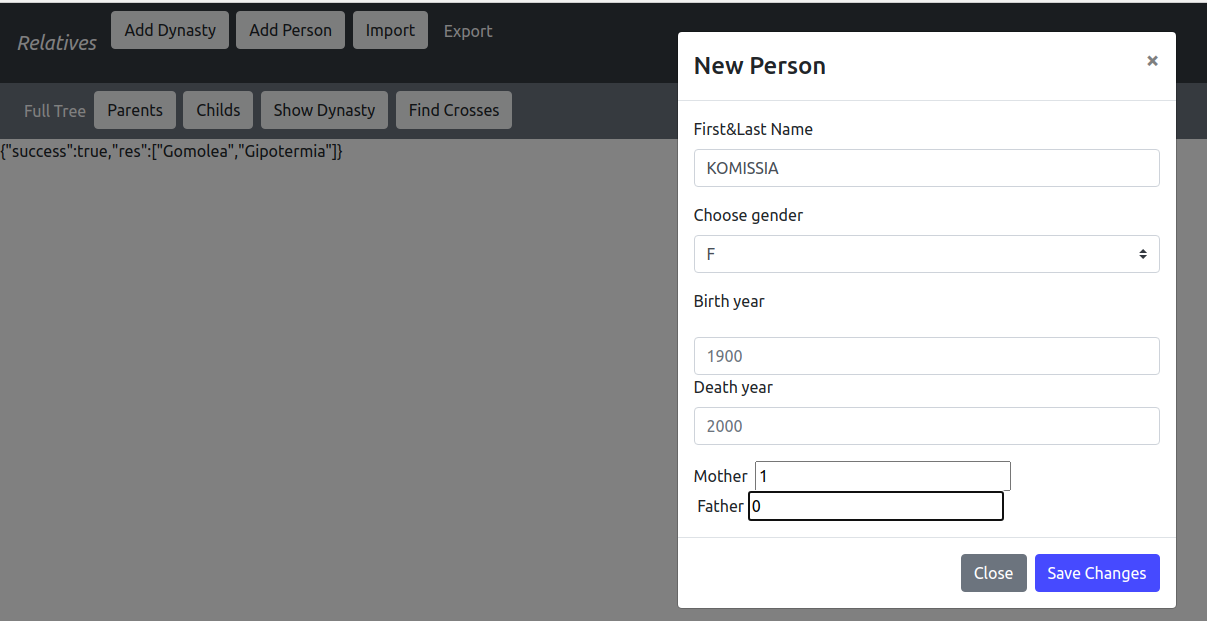
1. **РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ**

**Добавление династий**

****

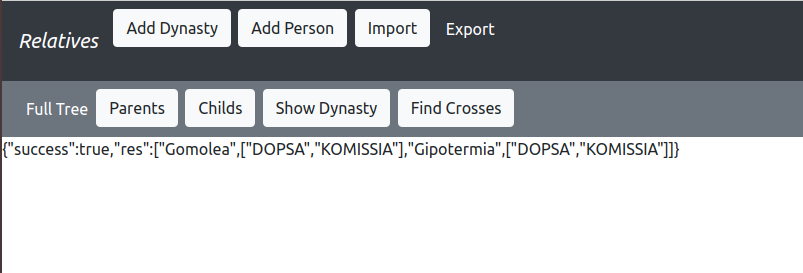
добавлено 2 династии, выведено 2 основателя

**Добавление человека**

****

добавим двух детей

добавим 2-х детей:

****

**ЗАКЛЮЧЕНИЕ**

Удалось сделать прототип, реализующий на backend все указанные операции, но не удалось полностью связать его с frontend. Из того, что удалось – export дерева, т.е вывод дерева из базы данных в JSON-представлении.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. <https://docs.mongodb.com/manual/reference/bson-types/>

2. <https://www.tutorialspoint.com/mongodb/mongodb_datatype.htm>

3. <https://neo4j.com/docs/2.1.5/graphdb-neo4j-properties.html>

4. Документация React-js: https://ru.reactjs.org/docs/getting-started.html

5. Документация Docker: https://dker.ru/docs/