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

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 с использованием СУБД Mongo.

Frontend – JS, с использованием React и Nginx.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

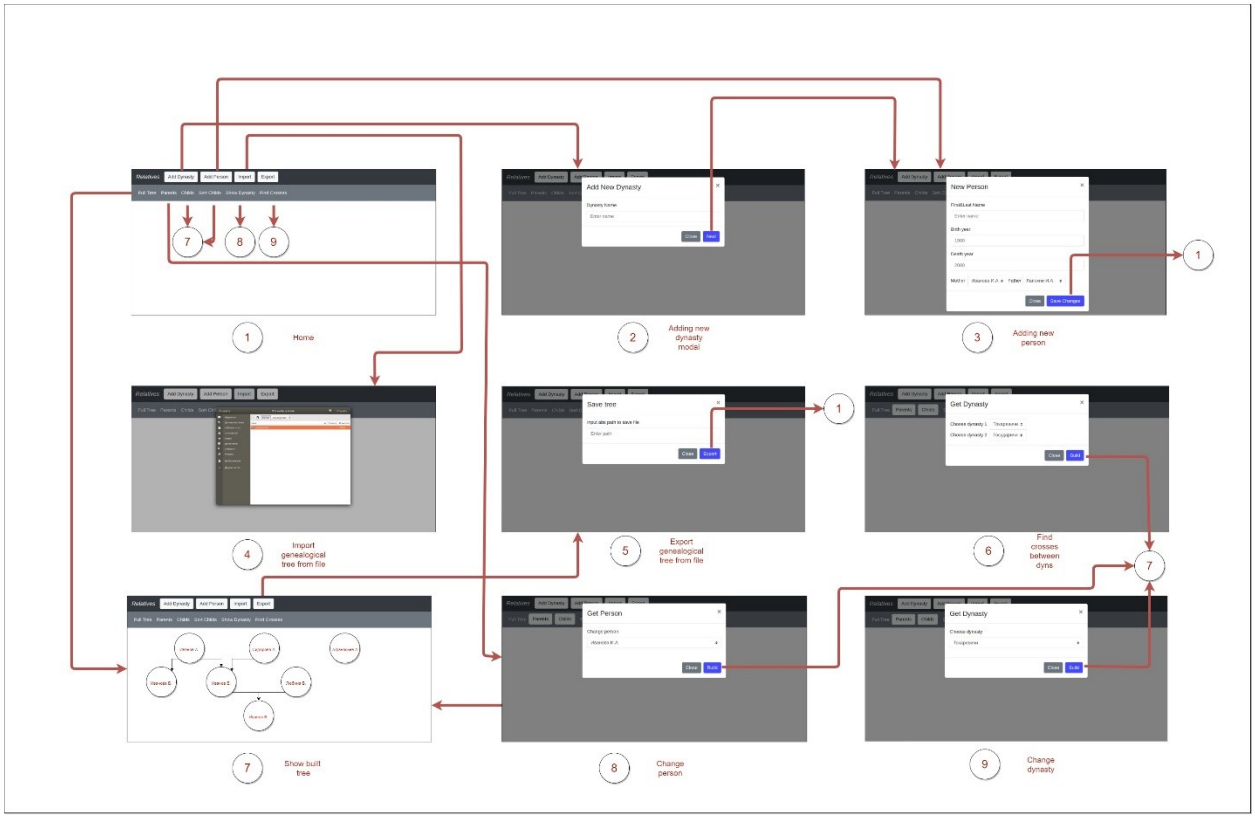
Веб-приложение с некоторым UI, который позволяет:

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

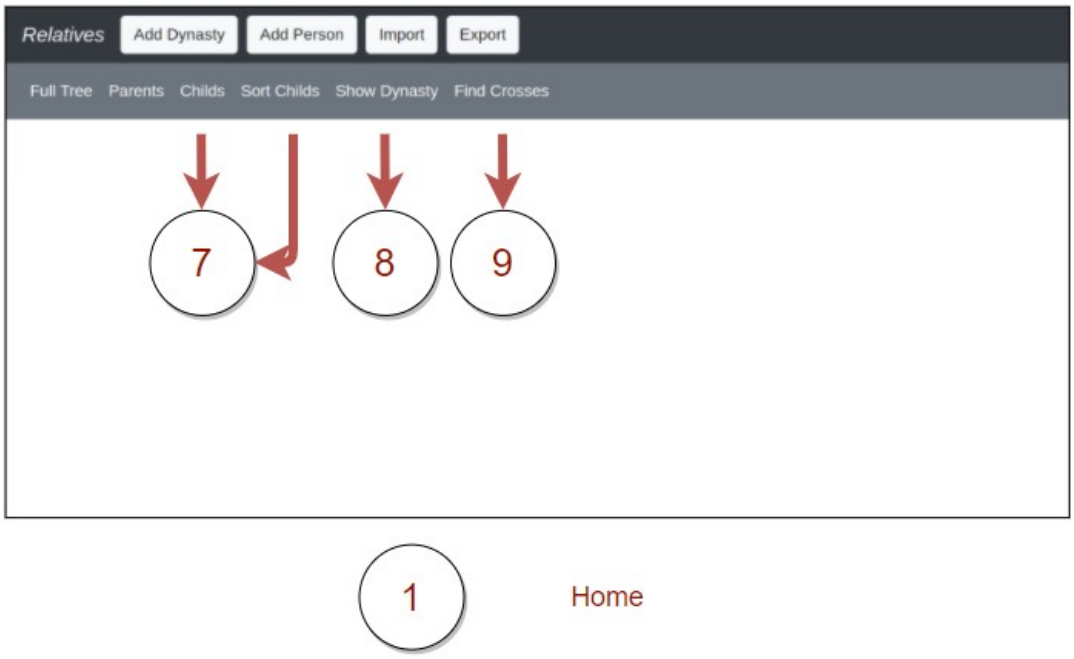
Использование Mongo DB в качестве СУБД.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

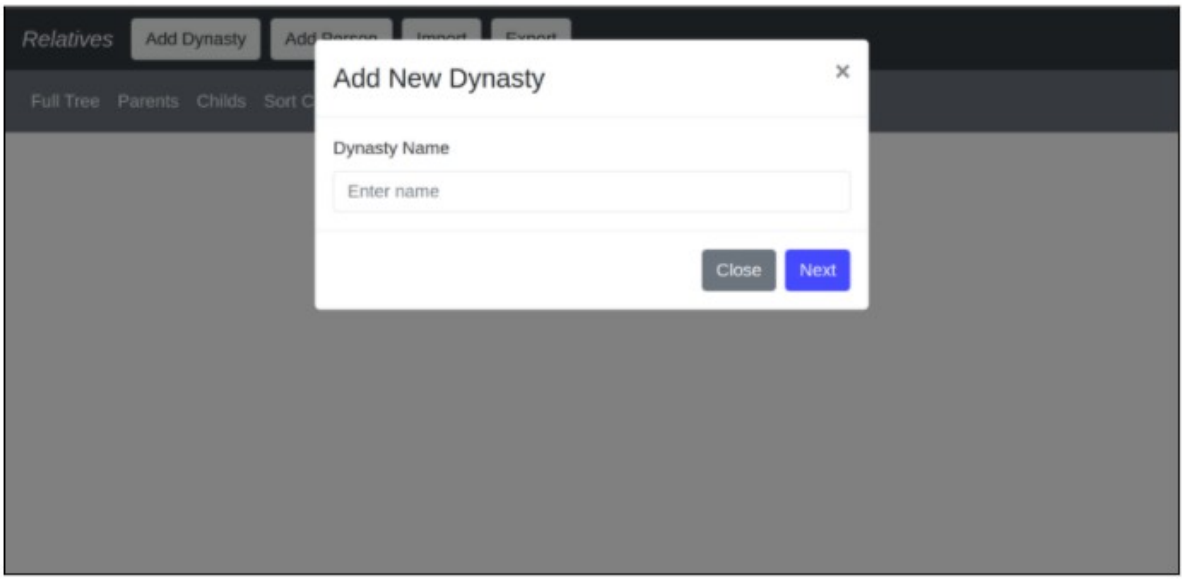
2.1. Макет пользовательского интерфейса



1.Домашняя страница:

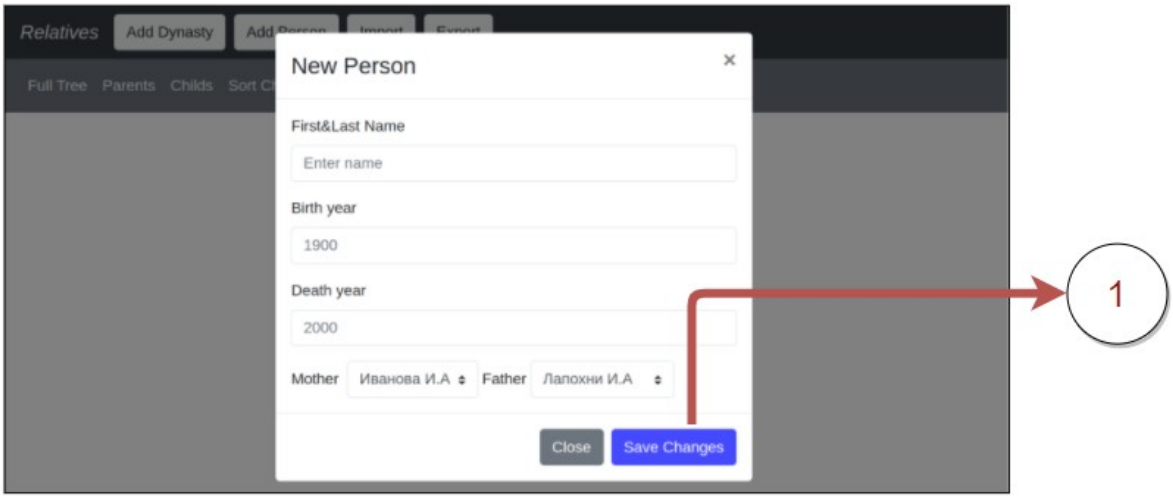


2. Добавление новой династии:



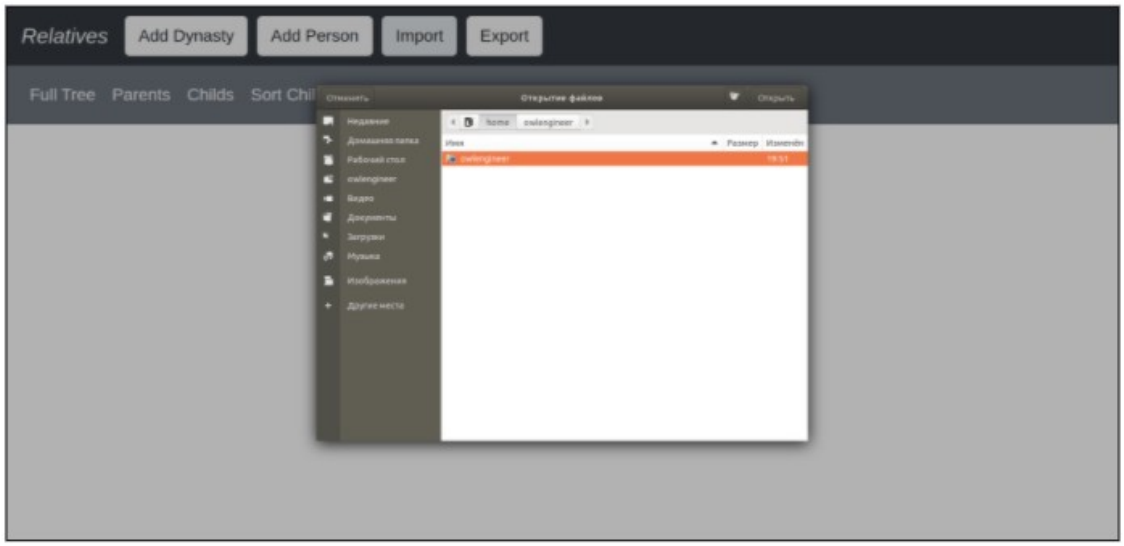
2 Adding new dynasty modal

3. Добавление нового человека:



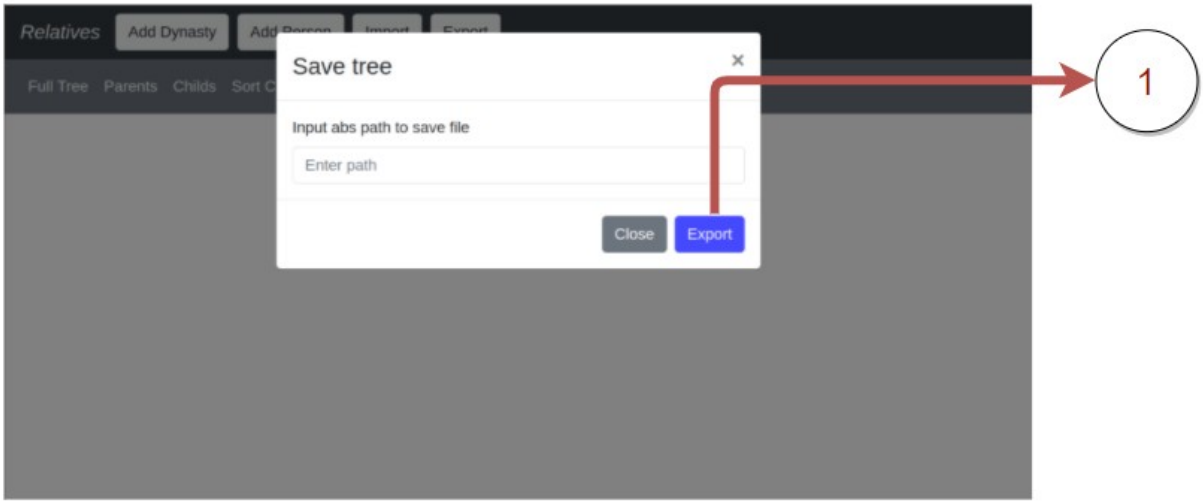
3 Adding new person

4. Импортировать дерево из файла:



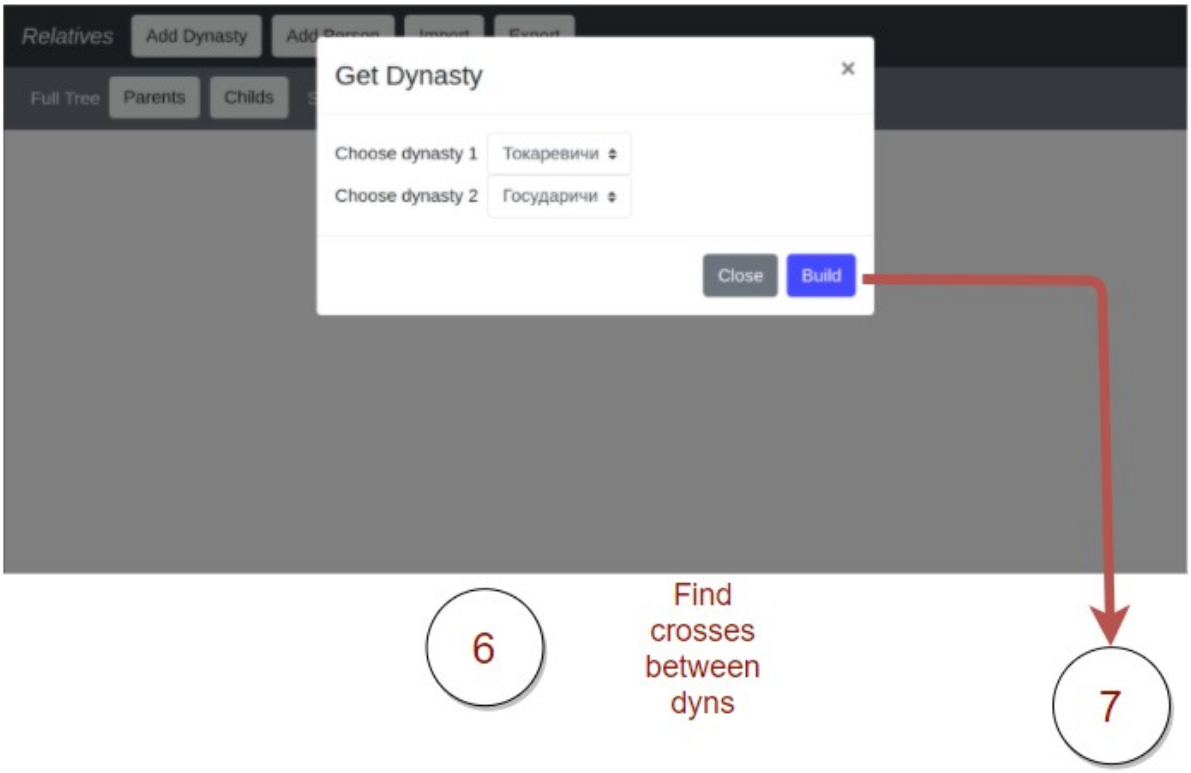
4 Import genealogical tree from file

5. Сохранить генеалогическое дерево в файл:

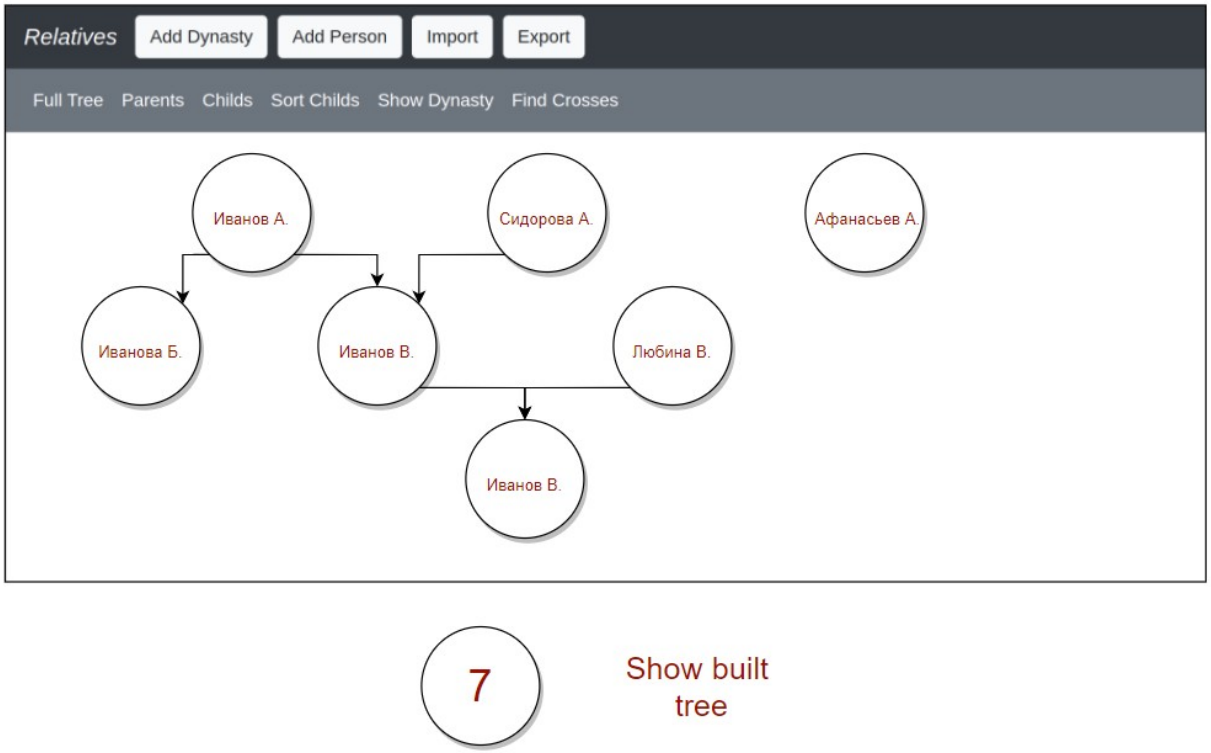


5 Export genealogical tree from file

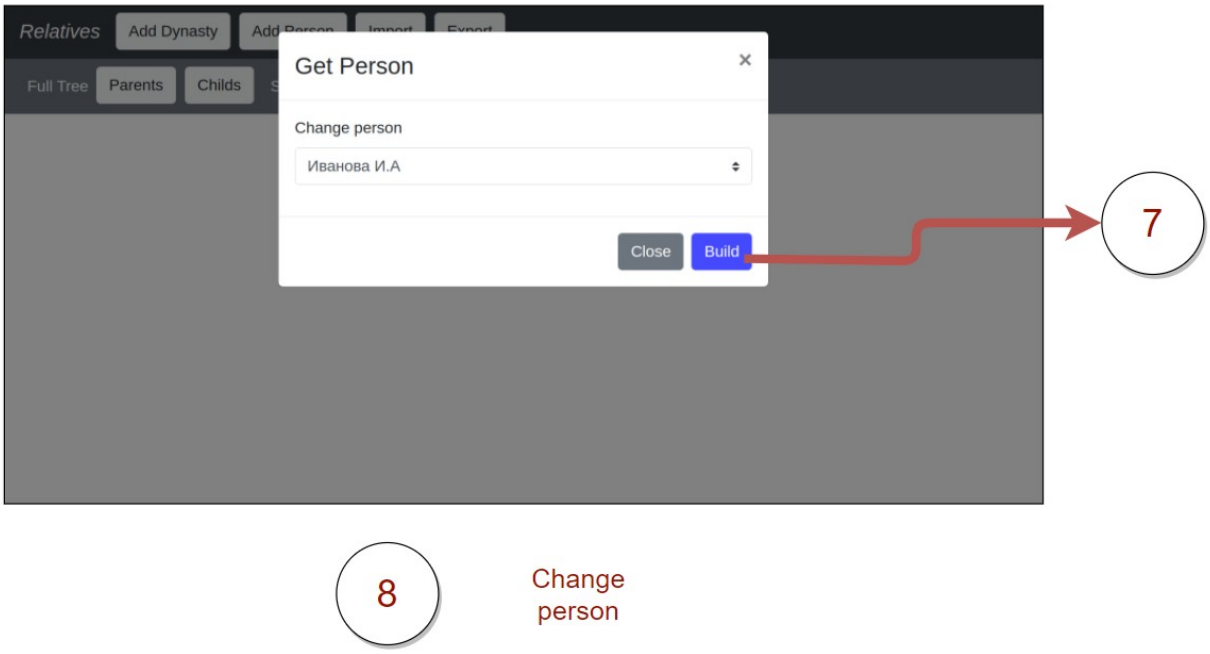
6. Найти пересечения между династиями:



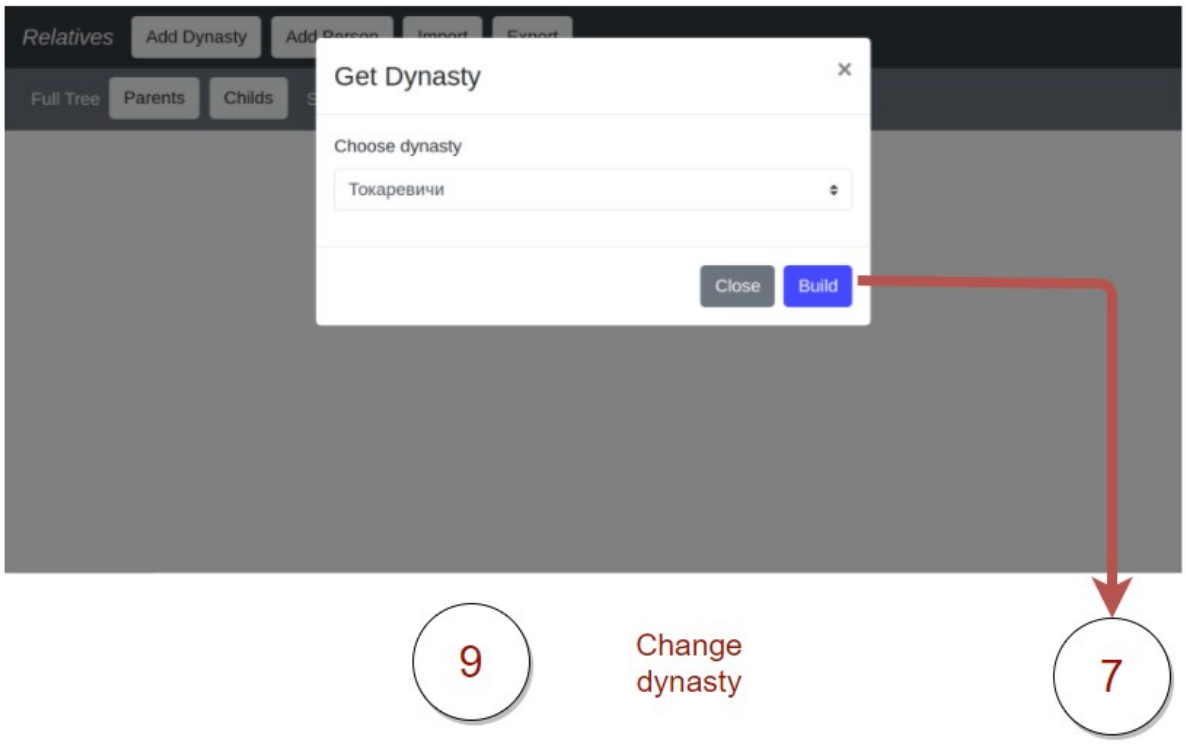
7. Просмотреть построенное дерево:



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:

   a. Дерева от детей и ниже

   b. Дерево предков

   c. Всего дерева целиком

4. Каждое из деревьев можно показать в режимах:

   a. без сортировки

   b. сортировка детей по старшинству слева направо (возраст определяется как разница текущей даты и даты рождения)

5. Отображение на графе:

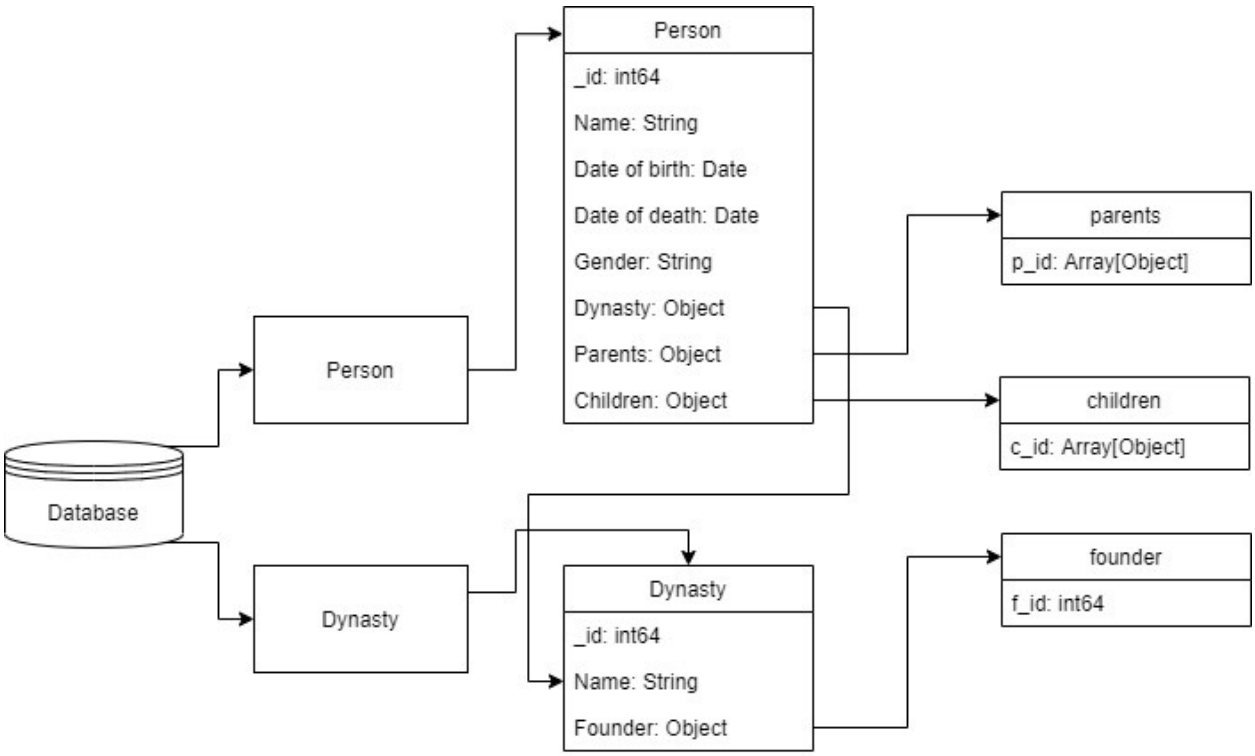
   a. Всю династию (для каждой династии отдельно)

   b. Две любые династии и поиск пересечений (общих детей)

3. 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
    - c_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.

$$V = 8N + 50N + 64N + N + 24N + 24N + 12N + \frac{N}{5}(8 + 50 + 12) = 197Nb$$

$$N = 16$$

$$V_{ф} = 1705b$$

$$Избыточность = \frac{197 \cdot 16}{1705} = 1.84$$

Примеры запросов:

Добавить человека

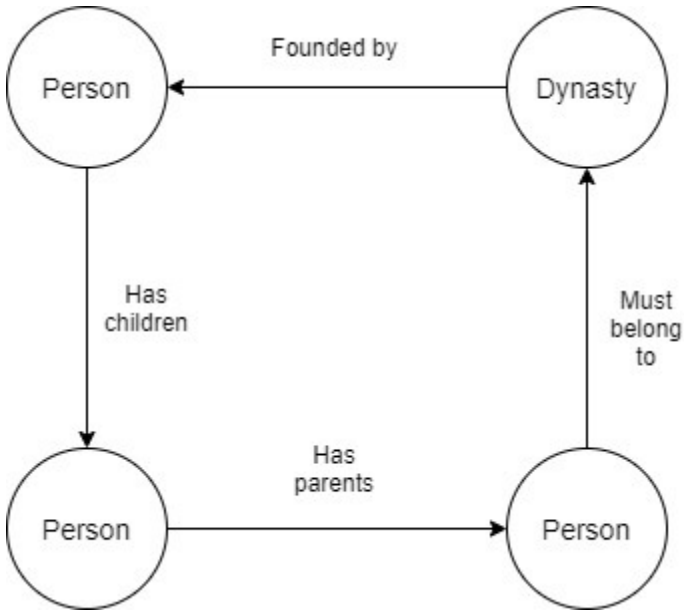
```
db.person.insert_many([{
'_id' : id,
'Name' : Name,
'Date of birth' : 12.12.1200,
'Date of death' : 24.05.1245,
```

```
'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)

2. Dynasty:

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

Оценка объема:

$$V = 2N + 50N + 100N + 2N + \frac{N}{5}(2 + 50) = 164,4Nb$$

$$N = 16$$

$$V = 2630.4b$$

Примеры запросов:

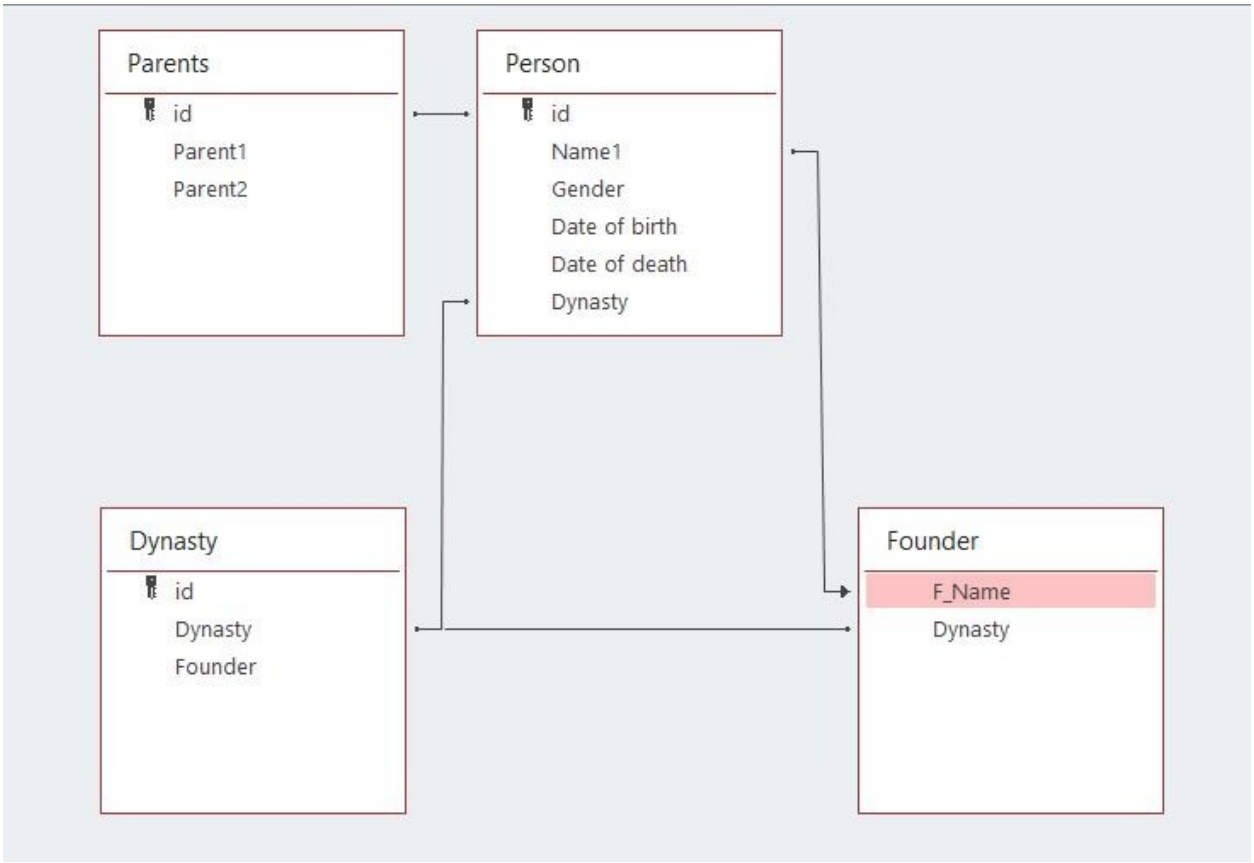
Добавить человека:

```
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)

2. Dynasty

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

3. Founder

- F_Name - имя основателя (короткий текст)
- Dynasty - (короткий текст) - название династии

4. Parents

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

Оценка объема:

V = 4*N* + *50*N + N + 8*2*N + 50*N* + *4*N/5 + 50*N/5* +*50*N/5 + 50*N/5* + *50*N/5 + 4*N* +*100*N = 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.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

ЗАКЛЮЧЕНИЕ

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/