МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Нереляционные базы данных»

Тема: ИС для бердвотчинга (Neo4j)

Студент гр. 7383 _____ Лосев М.Л.

Студент гр. 7383 _____ Кирсанов А.Я.

Студентка гр. 7383 _____ Иолшина В.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2020

ЗАДАНИЕ

НА ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

Студент Лосев М.Л.

Студент Кирсанов А.Я.

Студентка Иолшина В.

Группа 7383

Тема работы: ИС для бердвотчинга (Neo4j).

Исходные данные:

Требуется разработать приложение для бердвотчинга с использованием СУБД Neo4j.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Выводы»

«Приложения»

«Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 25 страниц.

Дата выдачи задания: 18.09.2020

Дата сдачи реферата:

Дата защиты реферата:

Студент	_____	Лосев М.Л.
Студент	_____	Кирсанов А.Я.
Студентка	_____	Иолшина В.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В рамках данного курса требовалось разработать приложение с использованием нереляционной базы данных на одну из предложенных тем. В качестве темы создаваемого приложения был выбран бердвотчинг, в качестве нереляционной системы управления базами данных – Neo4j. Исходный код приложения доступен в репозитории по ссылке [4].

SUMMARY

As part of this course, it was required to develop an application using a non-relational database for one of the proposed topics. Birdwatching was chosen as the topic of the application, and Neo4j as a non-relational database management system. The source code of the application is available in the repository at [4].

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – создание приложения для бердвотчинга, позволяющего добавлять в базу данных увиденную пользователем птицу, её фотографию и местоположение, а также просматривать добавленных другими пользователями птиц. Выбранные технологии решения включают в себя язык программирования Python 3 и набор библиотек PyQt5.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать приложение, в функциональность которого входят добавление новых элементов в базу данных, возможность импорта и экспорта базы данных, просмотр содержимого базы данных, а также фильтрация данных и статистика. В качестве системы управления базами данных требуется использовать Neo4j.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет UI

2.1.1. Главное окно (рис. 2.1)

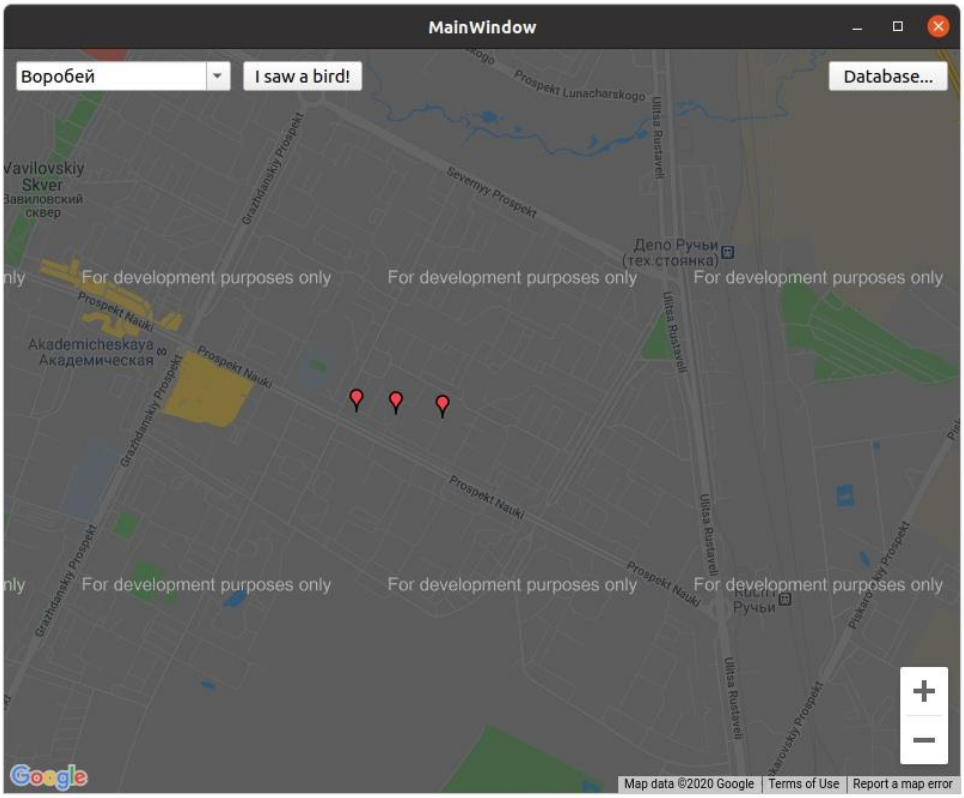

Рисунок 2.1 – Главное окно

2.1.2. Выбор распределения птиц определенного вида (рис. 2.2)

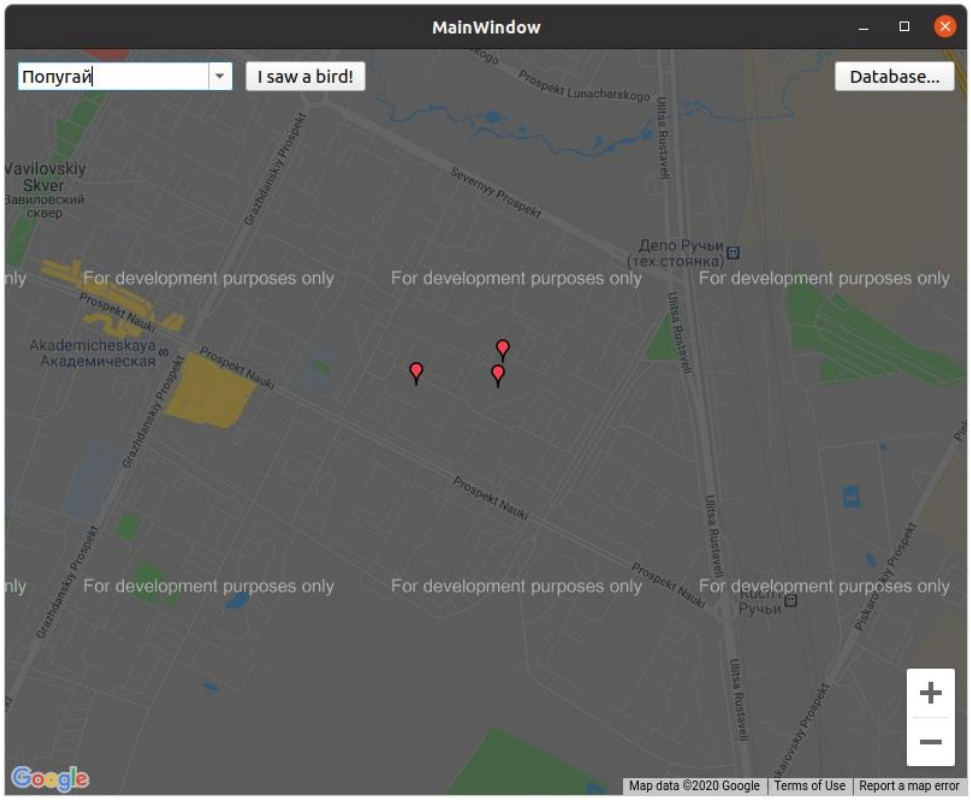

Рисунок 2.2 – Выбор распределения птиц определенного вида

2.1.3. Выбор маркера места, где была увидена птица (рис. 2.3)

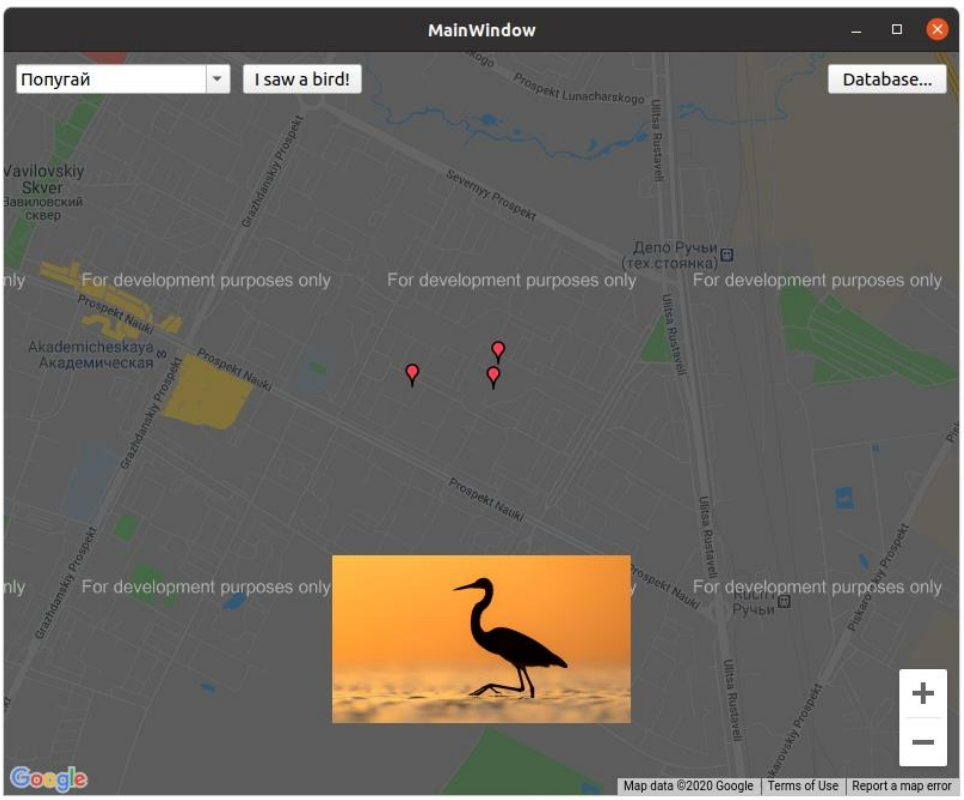

Рисунок 2.3 – Выбор маркера места, где была увидена птица

2.1.4. Окно добавления птицы (рис. 2.4)

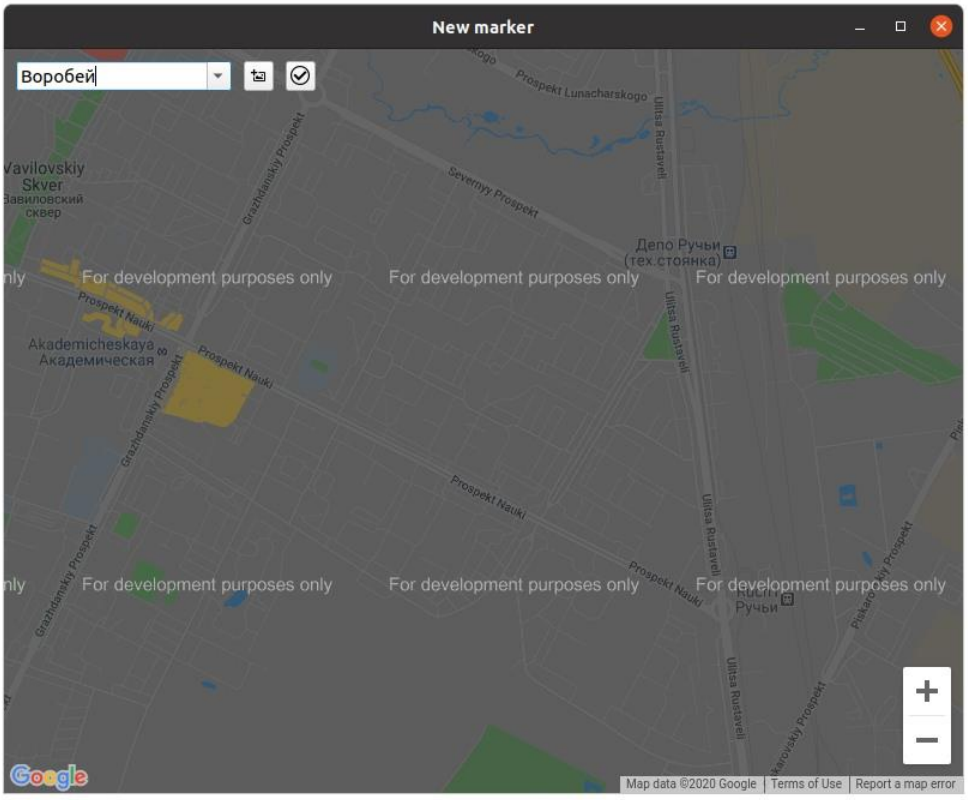

Рисунок 2.4 – Окно добавления птицы

2.1.5. Выбор вида увиденной птицы (рис. 2.5)

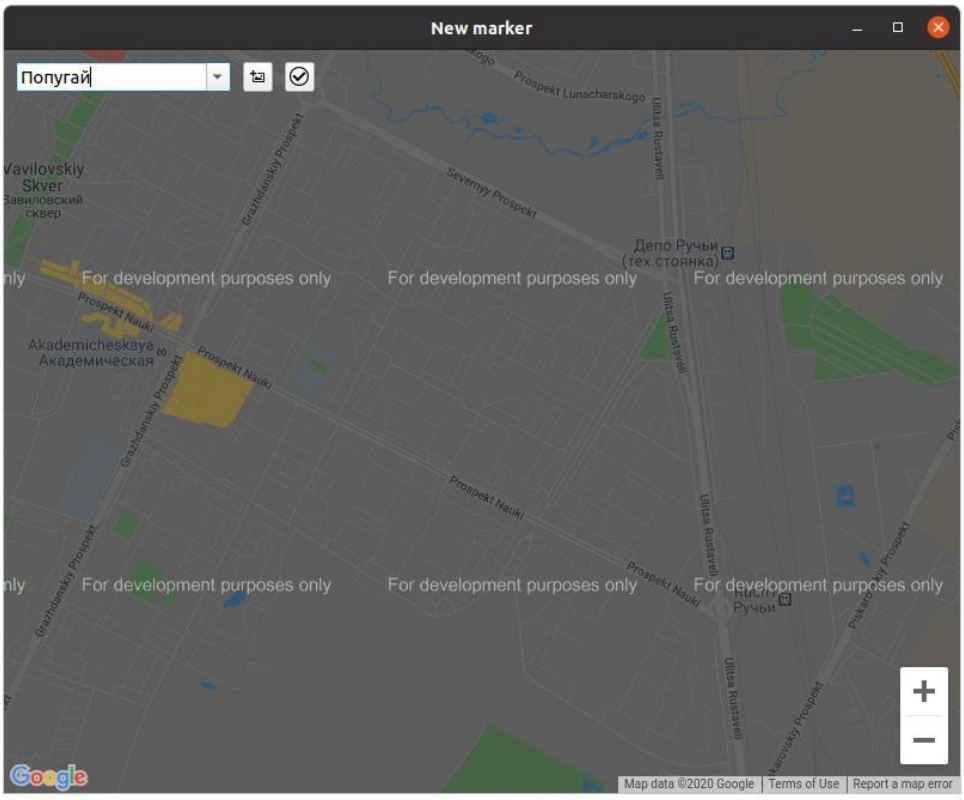

Рисунок 2.5 – Выбор вида увиденной птицы

2.1.6. Выбор места, где была увидена птица (рис. 2.6)

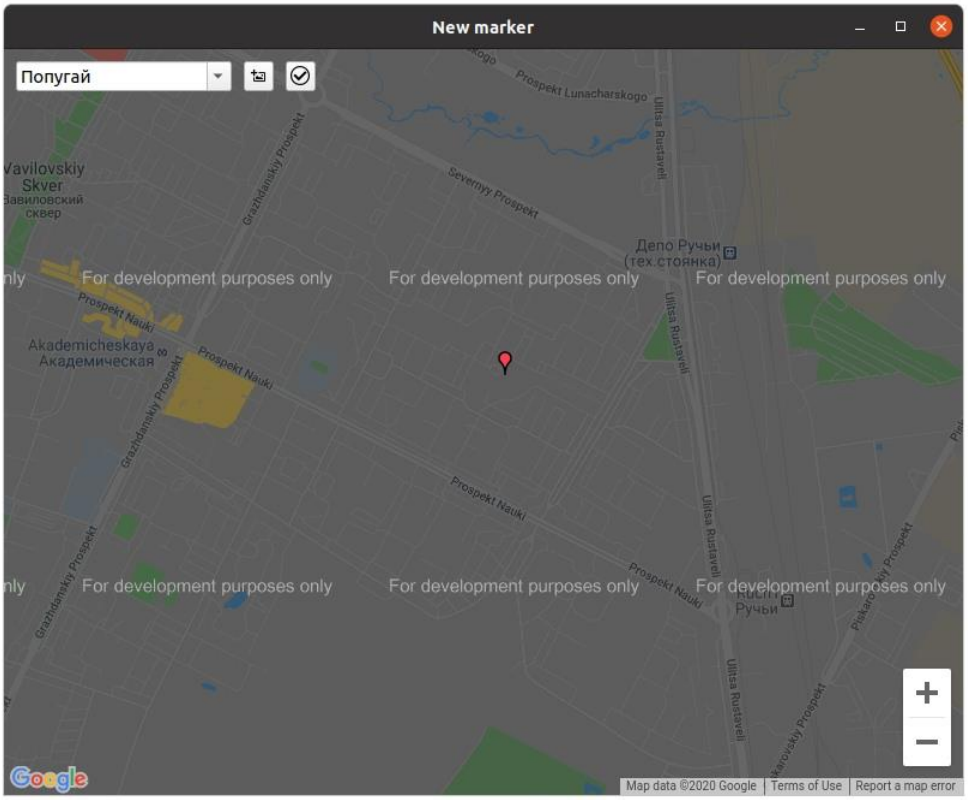

Рисунок 2.6 – Выбор места, где была увидена птица

2.1.7. Выбор файла с изображением увиденной птицы (рис. 2.7)

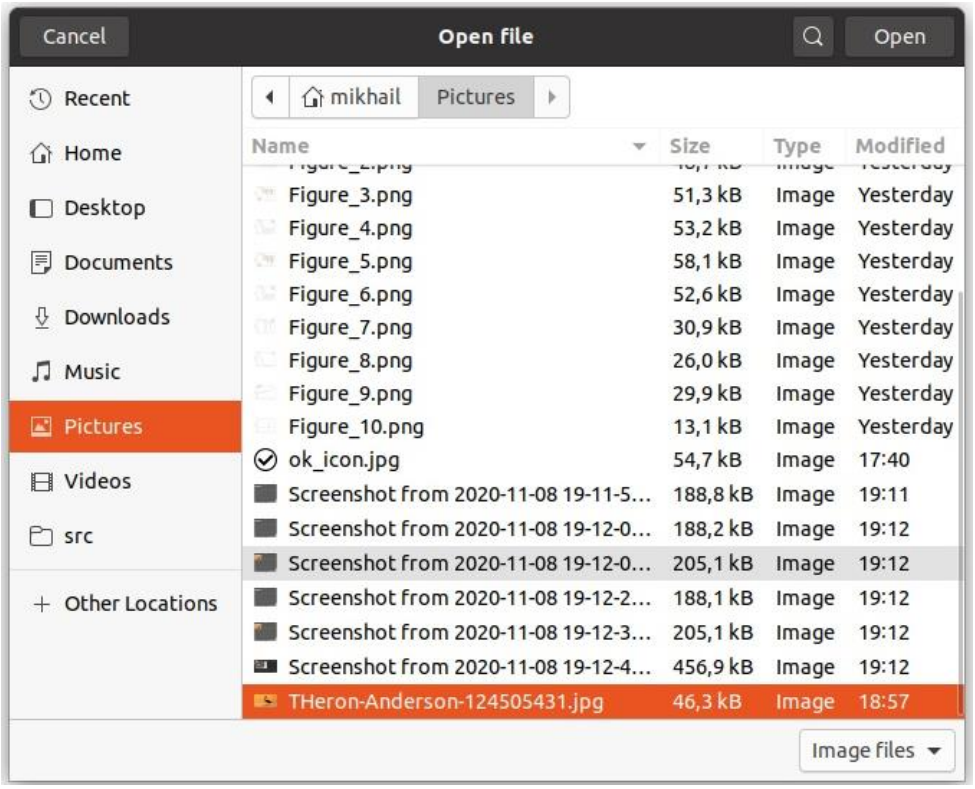

Рисунок 2.7 – Выбор файла с изображением увиденной птицы

2.1.8. Добавление файла с изображением увиденной птицы (рис. 2.8)

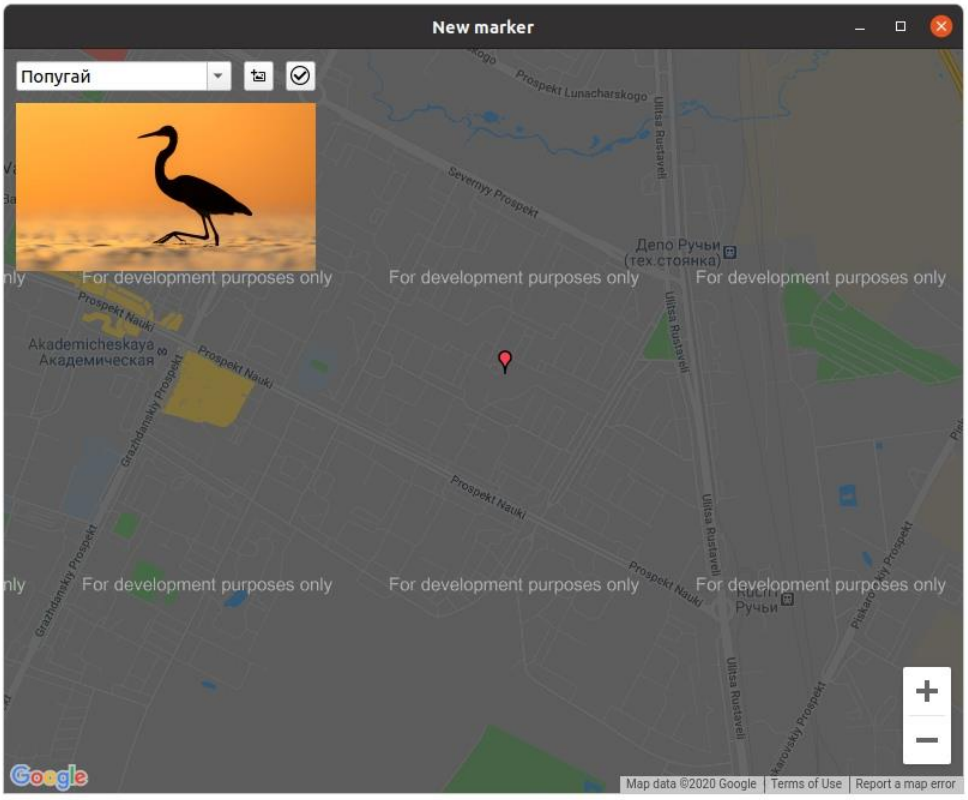

Рисунок 2.8 – Добавление файла с изображением увиденной птицы

2.1.9. Добавление маркера на карту (рис. 2.9)

Рисунок 2.9 – Добавление маркера на карту

2.1.10. Просмотр статистики и управление резервным копированием (рис. 2.10)

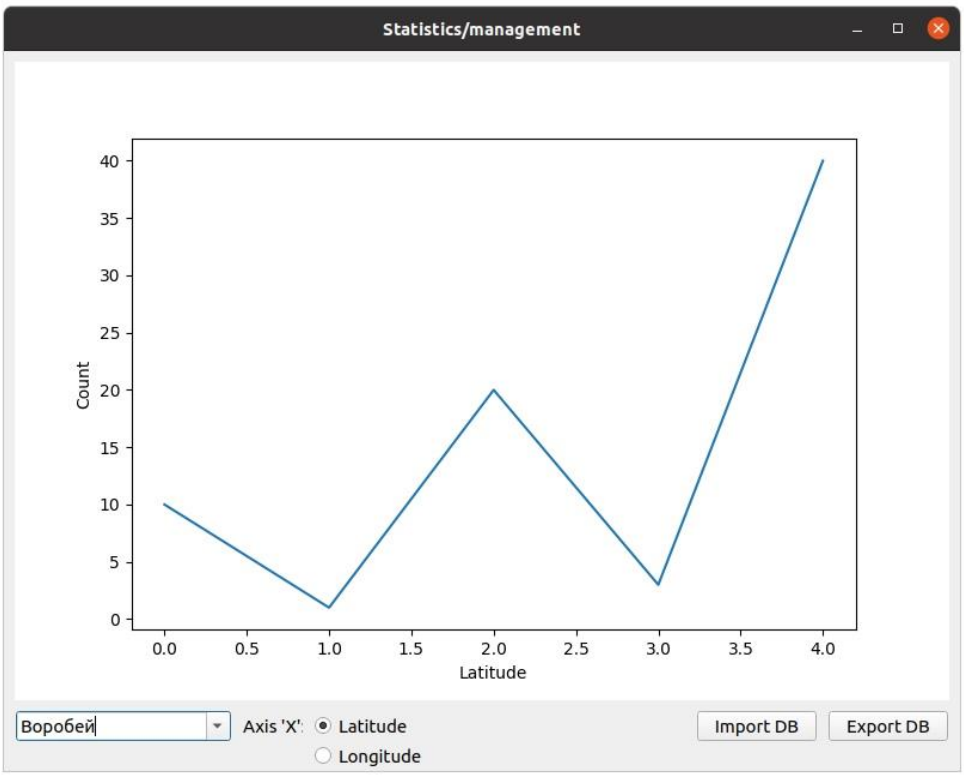

Рисунок 2.10 – Просмотр статистики и управление резервным копированием

2.1.11. Импорт базы данных (рис. 2.11)

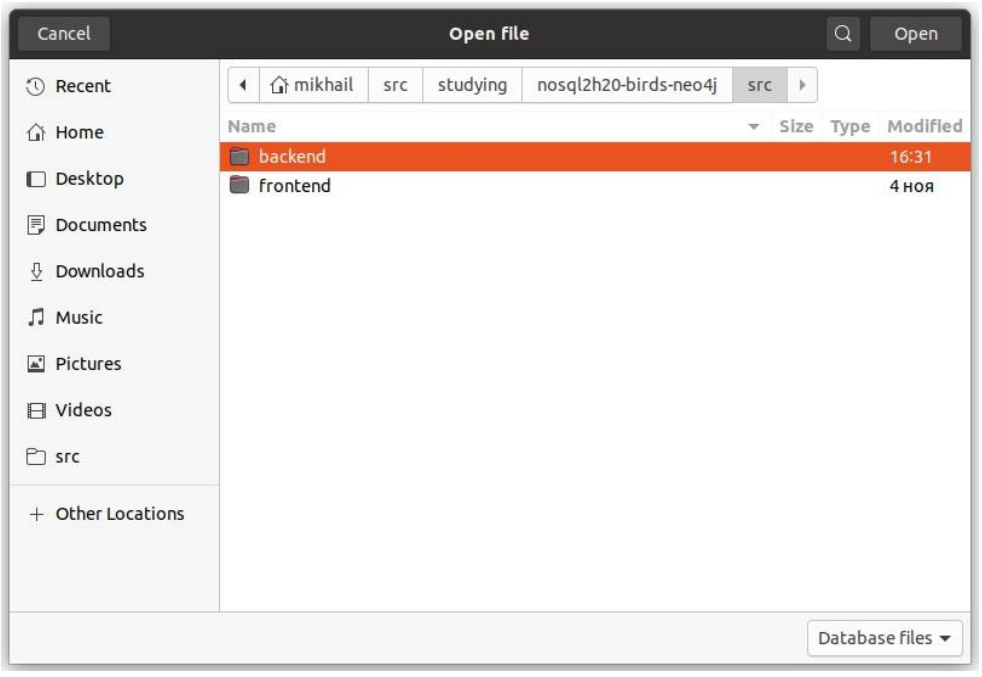

Рисунок 2.11 – Импорт базы данных

2.1.12. Экспорт базы данных (рис. 2.12)

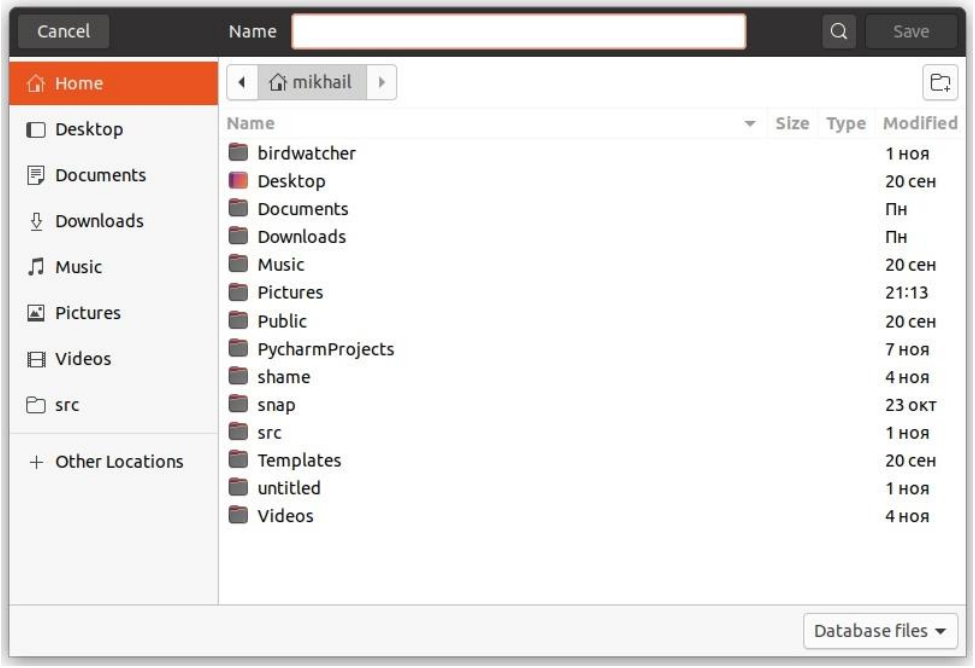

Рисунок 2.12 – Экспорт базы данных

2.1.13. Сохранение базы данных (рис. 2.13)

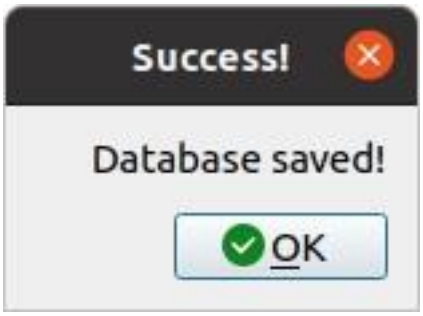

Рисунок 2.13 – Сохранение базы данных

2.2. Сценарии использования для задачи

Основной сценарий: «Добавление птицы в базу данных»

Действующее лицо: Пользователь

1. Пользователь хочет добавить найденную птицу и нажимает на кнопку "I saw a bird!".

2. Появляется экран добавления птицы с картой, полем для ввода названия этой птицы и кнопкой для загрузки файла с фотографией.

3. Пользователь хочет добавить название птицы.

i. Если вид птицы уже присутствует в базе данных, то пользователь выбирает его из списка.

ii. Если вид птицы отсутствует в базе данных, то пользователь вводит название в поле для ввода.

4. Пользователь ставит маркер на место на карте, в котором он нашел птицу.

5. Пользователь хочет загрузить фотографию и нажимает на кнопку загрузки файла.

6. Появляется окно выбора файла, где пользователь выбирает фотографию, которую он хочет загрузить со своего устройства.

7. Пользователь нажимает кнопку добавления птицы.

8. Появляется окно с сообщением о добавлении найденной птицы в базу данных.

9. Пользователь закрывает окно и может вернуться к пункту 1.

Дополнительный сценарий: «Просмотр найденных другими пользователями птиц по их видам»

Действующее лицо: Пользователь

1. Пользователь выбирает в списке вид птиц, места обитания которых он хочет посмотреть.

2. На карте остаются только маркеры выбранного вида птиц.

3. Пользователь выбирает маркер на карте, по которому он хочет посмотреть фотографию найденной в этом месте птицы.

4. Появляется фотография этой птицы.

5. Пользователь закрывает фотографию и может вернуться к пункту 1.

3. МОДЕЛЬ ДАННЫХ

3.1. Нереляционная модель Neo4j

3.1.1. Графическое представление

Графическое представление нереляционной модели Neo4j представлено на рис. 3.1.

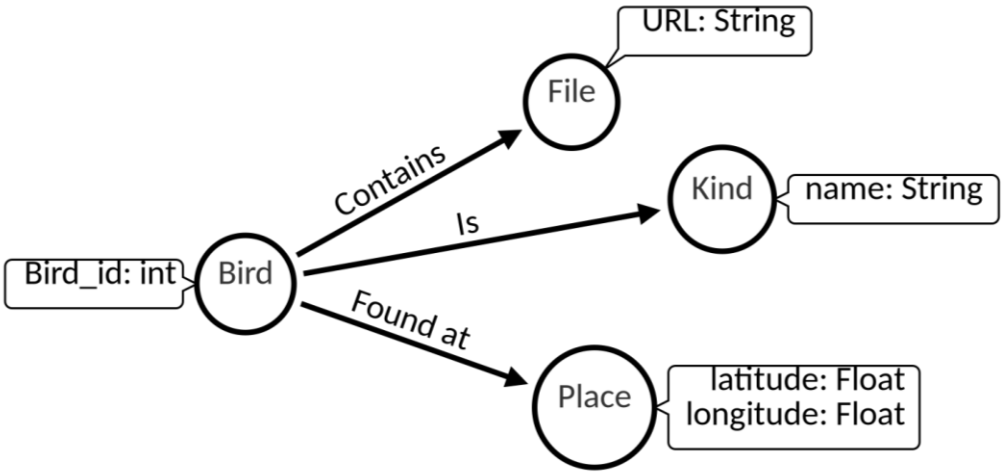

Рисунок 3.1 – Графическое представление модели БД

Разработанная модель включает в себя следующие сущности: Bird, File, Kind, Place.

3.1.2. Описание назначений коллекций, типов данных и сущностей

Сущность Bird содержит следующее свойство:

- bird_id – идентификатор птицы. Тип данных – Integer.

Сущность File содержит следующее свойство:

- URL – ссылка на файл с фотографией. Тип данных – String.

Сущность Kind содержит следующее свойство:

- name – название вида птицы. Тип данных – String.

Сущность Place содержит следующие свойства:

- latitude – широта координаты. Тип данных – Integer.
- longtitude – долгота координаты. Тип данных – Integer.

3.1.3. Оценка удельного объема информации, хранимой в модели

Bird: bird_id – 4B

File: URL – 200B

Kind: name – 100B

Place: latitude – 4B, longtitude – 4B

Общий размер – 312B.

Пусть N – количество птиц. Тогда «чистый» объем равен N * 312B. Фактический объем равен N * 3 * 34B * 312B = N * 31824B.

Избыточность модели: (N * 31824B) / (N * 312B).

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

- Выбор всех видов птиц

```
MATCH (p:Kind)
RETURN p
```

- Выбор птиц по виду

```
MATCH (id:Bird)-[:Is]->(Kind {name: "Грач"})
RETURN id
```

- Выбор конкретных птиц

```
MATCH (a:Bird {Bird_id: 1})
RETURN a
```

- Выбор всех птиц

```
MATCH (a:Bird)
RETURN a
```

- Просмотр ареала птиц конкретного вида

```
MATCH (a:Bird)-[:Found_at]->(b:Place)
RETURN b
```

3.2. Нереляционная модель MongoDB

3.2.1. Схема базы данных

Схема нереляционной модели MongoDB представлена на рис. 3.2.

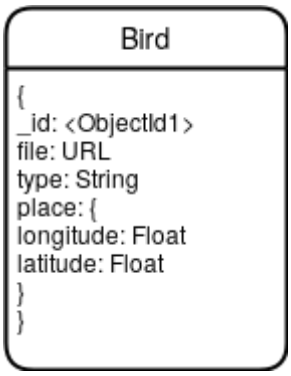

Рисунок 3.2 – Схема нереляционной базы данных

3.2.2. Описание назначений коллекций, типов данных и сущностей

Описание полей коллекции Bird:

- id – идентификатор. Тип данных: ObjectId.
- file – файл с фотографией. Тип данных: String.
- type – название вида птицы. Тип данных: String.
- place – место на карте.
    - latitude – широта координаты. Тип данных: Integer.
    - longtitude – долгота координаты. Тип данных: Integer.

3.2.3. Оценка удельного объема информации, хранимой в модели Bird:

id – 12B

file – 200B

type – 100B

latitude – 4B

longtitude – 4B

Общий размер – 312B.

Пусть N – количество птиц. Тогда «чистый» объем равен N * 320B. Фактический объем равен чистому N * 320B.

Избыточность в данной модели отсутствует.

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

- Выбор всех видов птиц

```
db.collection.distinct("type")
```

• Выбор птиц по виду

```
db.collection.find({type: "Грач"})
```

• Выбор конкретных птиц

```
db.collection.find({id: "1"})
```

• Выбор всех птиц

```
db.collection.find({ })
```

• Просмотр ареала птиц конкретного вида

```
db.collection.find({place: {type: "Сорока"}})
```

3.3. Реляционная модель SQL

3.3.1. Графическое представление

Графическое представление реляционной модели представлено на рис. 3.3.

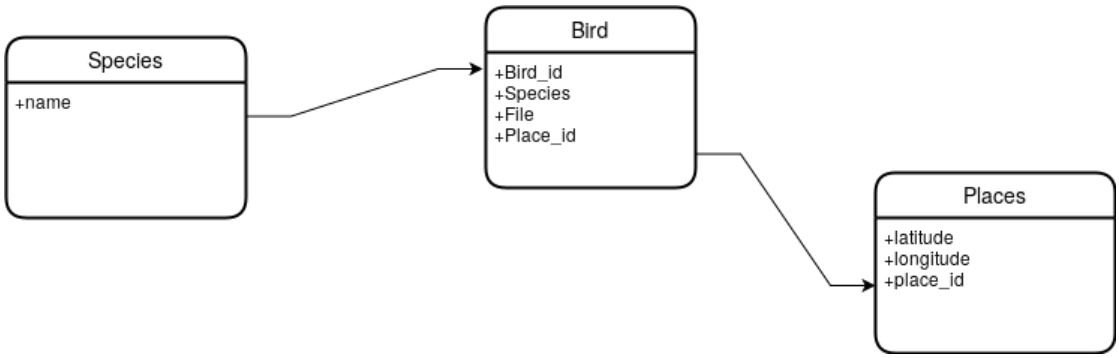

Рисунок 3.3 – Графическое представление модели БД

Разработанная модель включает в себя следующие сущности: Bird, Species, Places.

3.3.2. Описание назначений коллекций, типов данных и сущностей

Сущность Bird содержит следующие свойства:

- bird_id – идентификатор птицы. Тип данных – Integer.
- Species – название вида птицы. Тип данных – String.
- File – файл с фотографией. Тип данных – String.
- Place_id – идентификатор места на карте. Тип данных – Integer.

Сущность Species содержит следующее свойство:

• name – название вида птицы. Тип данных – String.

Сущность Places содержит следующие свойства:

• latitude – широта координаты. Тип данных – Integer.

• longtitude – долгота координаты. Тип данных – Integer.

• place_id – идентификатор места на карте. Тип данных – Integer.

3.3.3. Оценка удельного объема информации, хранимой в модели

Bird:

bird_id – 4B

Species – 100B

File – 200B

Place_id – 4B

Species:

name – 100B

Places:

latitude – 4B

longtitude – 4B

place_id – 4B

Общий размер – 408B.

Пусть N – количество птиц. Тогда «чистый» объем равен N * 408B. Фактический объем равен N * 420B.

Избыточность модели: (N * 420B) / (N * 408B).

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

• Выбор всех видов птиц

```
SELECT Spec
FROM Species;
```

• Выбор птиц по виду

```
SELECT Bird.Bird_id
FROM Bird WHERE (((Bird.[Species])="Грач"));
```

• Выбор конкретных птиц

```
SELECT *
FROM Bird WHERE Bird_id=1;
```

- Выбор всех птиц

```
SELECT *
FROM Bird;
```

- Просмотр ареала птиц конкретного вида

```
SELECT latitude, longitude
FROM (Species INNER JOIN Bird ON Bird.Species=Species.Spec)
INNER JOIN Places ON Places.Place_id=Bird.place_id;
```

3.4. Сравнение моделей

При расчете объема данных избыточность модели Neo4j получилась много выше, чем избыточность SQL, у MongoDB избыточность отсутствует. При этом запросы Neo4j и MongoDB компактнее и их выполнение занимает меньше времени. Из всего вышесказанного можно сделать вывод, что MongoDB для данной задачи подходит больше.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Схема экранов приложения

Схема экранов приложения приведена на рис. 4.1.

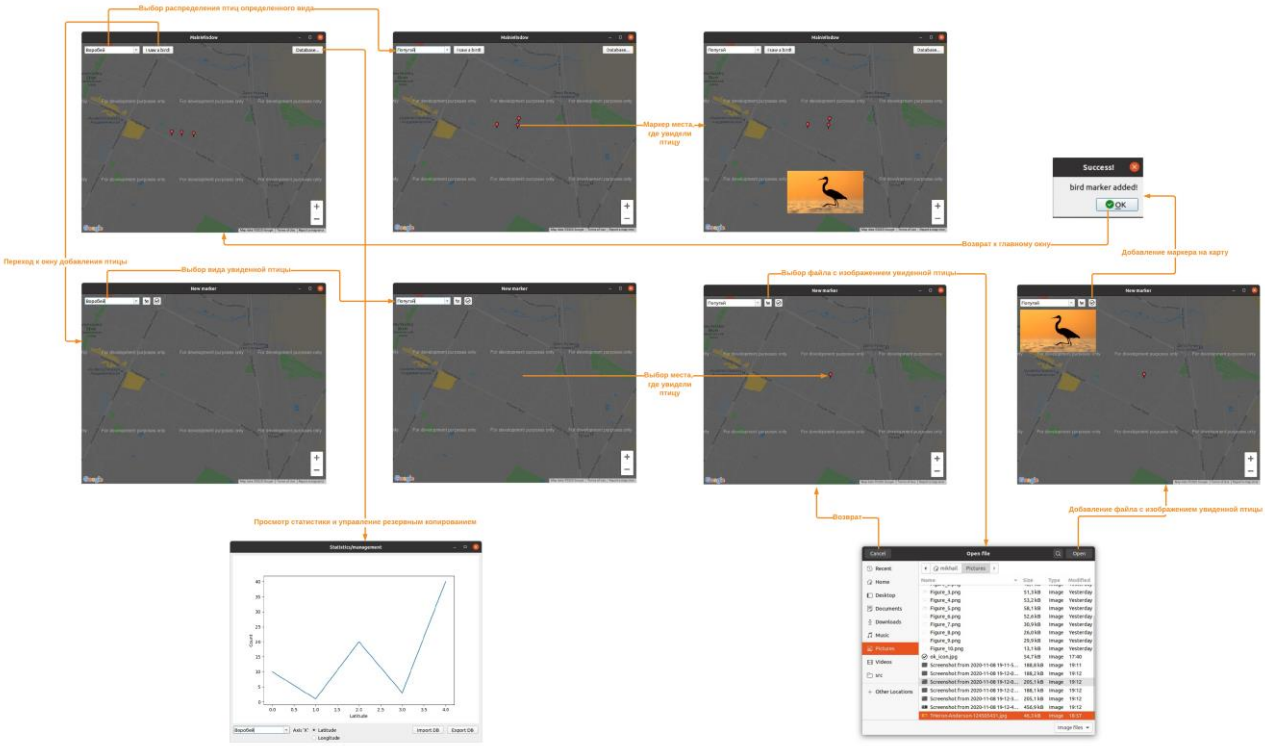

Рисунок 4.1 – Схема экранов приложения

4.2. Использованные технологии

Neo4j, Python 3, PyQt5, Docker, Javascript, html, json.

4.3. Ссылки на приложение

Github: https://github.com/moevm/nosql2h20-birds-neo4j

5. ВЫВОДЫ

5.1. Достигнутые результаты

Было разработано приложение для бердвотчинга, позволяющее добавлять в базу данных увиденную пользователем птицу, её фотографию и местоположение, а также просматривать добавленных другими пользователями птиц. В функциональность приложения входят добавление новых элементов в базу данных, возможность импорта и экспорта базы данных, просмотр содержимого базы данных, а также фильтрация данных и статистика.

5.2. Недостатки и пути для улучшения полученного решения

К недостаткам данной реализации можно отнести использование в ней локальной базы данных. Для улучшения предложенного решения можно отказаться от использования локальной базы данных и разместить её на удаленном сервере.

5.3. Будущее развитие решения

Возможности будущего развития решения предполагают использование облачной базы данных, а также разработку веб-приложения и приложения для Android и iOS.

6. ПРИЛОЖЕНИЯ

6.1. Документация по сборке и развертыванию приложения

Инструкция для Docker:

1. Скачать проект из репозитория [4].
2. Запустить ./runCompose.sh.

6.2. Пример файла, который ожидает система для импорта

Система ожидает для импорта файл в формате csv. Пример содержимого такого файла:

[{"name": "\u0412\u043e\u0440\u043e\u0431\u0435\u0439", "latitude": 60.0161740495795, "longitude": 30.413668497314465, "url": "/home/mikhail/Pictures/bird_photo1.jpg"}, {"name": "\u041f\u0435\u0442\u0443\u0445", "latitude": 60.00682175661092, "longitude": 30.411351068725597, "url": "/home/mikhail/Pictures/ok_icon.jpg"}]

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Neo4j // Neo4j Documentation. URL: https://neo4j.com/docs/ (дата обращения: 04.11.2020).

2. Документация Python 3.9.1 // Python 3.9.1 documentation. URL: https://docs.python.org/3/ (дата обращения: 04.11.2020).

3. Документация Docker // Docker documentation. URL: https://docs.docker.com/ (дата обращения: 04.11.2020).

4. Репозиторий Github // URL: https://github.com/moevm/nosql2h20-birds-neo4j