**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: **Система indoor**-**навигации / карт (MongoDB)**

|  |  |  |
| --- | --- | --- |
| Студент гр. 7383 |  | Власов Р.А. |
| Студентка гр. 7383 |  | Маркова А.В. |
| Студент гр. 7383 |  | Сычевский Р.А. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2020

**ЗАДАНИЕ**

**на Индивидуальное домашнее задание**

|  |  |  |
| --- | --- | --- |
| Студент Власов Р.А.  Студентка Маркова А.В.  Студент Сычевский Р.А. | | |
| Группа 7383 | | |
| Тема работы: Система indoor-навигации / карт (MongoDB) | | |
| Исходные данные:  Требуется разработать приложение для indoor-навигации с использованием СУБД MongoDB. | | |
| Содержание пояснительной записки:  «Введение»  «Качественные требования к решению»  «Сценарии использования»  «Модель данных»  «Разработанное приложение»  «Выводы»  «Приложение»  «Список использованных источников» | | |
| Предполагаемый объем пояснительной записки:  Не менее 25 страниц. | | |
| Дата выдачи задания: 18.09.2020 | | |
| Дата сдачи реферата: 29.12.2020 | | |
| Дата защиты реферата: 29.12.2020 | | |
| Студент гр. 7383 |  | Власов Р.А. |
| Студентка гр. 7383 |  | Маркова А.В. |
| Студент гр. 7383 |  | Сычевский Р.А. |
| Преподаватель |  | Заславский М.М. |

# Аннотация

В рамках данного курса требовалось разработать приложение, реализующее систему indoor-навигации, с использованием нереляционной базы данных MongoDB. Приложение было реализовано на языке JavaScript на платформе NodeJS с использованием фреймворка Express.

# Summary

As part of the course, it was required to develop an indoor-navigation system application using non-relational database management system MongoDB. The application was implemented in JavaScript on the NodeJS platform using the Express framework.

# Содержание

[Аннотация 4](#_Toc60087673)

[Summary 4](#_Toc60087674)

[Содержание 5](#_Toc60087675)

[1. Введение 6](#_Toc60087676)

[2. Качественные требования к решению 7](#_Toc60087677)

[3. Сценарий использования 8](#_Toc60087678)

[3.1. Макет пользовательского интерфейса 8](#_Toc60087679)

[3.2. Макет панели администратора 10](#_Toc60087680)

[3.3. Сценарии использования пользовательского интерфейса 13](#_Toc60087681)

[3.4. Сценарии использования панели администратора 14](#_Toc60087682)

[4. Модель данных 15](#_Toc60087683)

[4.1. Нереляционная модель MongoDB 15](#_Toc60087684)

[4.2. Нереляционная модель Neo4j 17](#_Toc60087685)

[4.3. Реляционная модель SQL 19](#_Toc60087686)

[4.4. Сравнение моделей 21](#_Toc60087687)

[5. Разработанное приложение 22](#_Toc60087688)

[5.1. Схема экранов приложения 22](#_Toc60087689)

[5.2. Использованные технологии 25](#_Toc60087690)

[5.3. Ссылки на приложение 26](#_Toc60087691)

[6. Выводы 27](#_Toc60087692)

[6.1. Достигнутые результаты 27](#_Toc60087693)

[6.2. Недостатки и пути для улучшения полученного решения 27](#_Toc60087694)

[6.3. Будущее развитие решения 27](#_Toc60087695)

[7. Приложения 28](#_Toc60087696)

[7.1. Документация по сборке и развертыванию приложения 28](#_Toc60087697)

[7.2. Пример файла для импорта 28](#_Toc60087698)

[Список использованных источников 29](#_Toc60087699)

# Введение

Целью работы является создание приложения для indoor-навигации, в функциональность которого входят импорт карт помещений, построение маршрутов между выбранными аудиториями, экспорт данных. Выбранный нами стек технологий включает в себя JavaScript, Node.js, Express.js, MongoDB, Pug, Less.

Для достижения поставленной цели требуется решить следующие задачи:

* Реализовать макет интерфейса;
* Определить сценарии использования;
* Реализовать макет данных;
* Настроить запуск приложения в docker-контейнере;
* Реализовать основной функционал приложения.

# Качественные требования к решению

Требуется разработать интуитивно понятный графический интерфейс для приложения, в функциональность которого должны входить:

* Главная страница, на которой содержится:
  + Карта помещений;
  + Интерфейс управления картой;
  + Интерфейс поиска аудиторий;
  + Интерфейс построения маршрутов.
* Страница администратора сайта, на которой содержится:
  + Интерфейс просмотра информации о помещениях;
  + Интерфейс импорта данных;
  + Интерфейс экспорта данных.

В качестве системы управления базами данных использовать MongoDB.

# Сценарий использования

## **Макет пользовательского интерфейса**

|  |
| --- |
| На рис. 1 изображено главное окно приложения.    Рисунок 1 – Главное окно |
| Поиск аудитории продемонстрирован на рис. 2.    Рисунок 2 – Поиск аудитории |
| После выбора необходимой аудитории открывается окно карточки, которое показано на рис. 3.    Рисунок 3 – Окно карточки |
| Заполнение поля «маршрут сюда» представлено на рис. 4.    Рисунок 4 – Заполнение поля «маршрут сюда» |
| Заполнение поля «маршрут отсюда» показано на рис. 5.    Рисунок 5 – Заполнение поля «маршрут отсюда» |
| После того как заполнены оба поля: «маршрут отсюда» и «маршрут сюда», между выбранными аудиториями строится маршрут до аудитории, продемонстрированный на рис. 6.    Рисунок 6 – Построение пути |

## Макет панели администратора

|  |
| --- |
| На рис. 7 изображено главное окно администратора приложения.    Рисунок 7 – Главное окно |
| Если при импорте происходит какая-либо ошибка, то программа выводит окно, представленное на рис. 8.    Рисунок 8 – Ошибка импорта |
| После успешного импорта файла появляется окно, продемонстрированное на рис. 9.    Рисунок 9 – Успешный импорт файла |
| Поиск аудитории представлен на рис. 10.    Рисунок 10 – Поиск аудитории |
| Окно карточки показано на рис. 11.    Рисунок 11 – Окно карточки |

## Сценарии использования пользовательского интерфейса

Основной сценарий: «Поиск маршрута»

Действующее лицо: Пользователь

1. Пользователь заходит на сайт приложения;
2. Пользователь вводит в поисковую строку нужную ему аудиторию;
3. Пользователь выбирает нужную аудиторию из списка предложенных;
4. На экране появляется карточка с информацией об аудитории;
5. Пользователь выбирает «маршрут сюда» или «маршрут отсюда»;
6. Если пункт отправления или пункт назначения не заполнен – вернуться к пункту 2;
7. Программа находит и отображает маршрут.

Дополнительный сценарий: «Поиск информации об аудитории»

Действующее лицо: Пользователь

1. Пользователь заходит на сайт приложения;
2. Пользователь вводит в поисковую строку нужную ему аудиторию;
3. Пользователь выбирает нужную аудиторию из списка предложенных;
4. На экране появляется карточка с информацией об аудитории.

## Сценарии использования панели администратора

Основной сценарий: «Изменение базы данных»

Действующее лицо: Администратор

1. Администратор заходит на сайт приложения;
2. Администратор перетаскивает на поле импорта или выбирает файл с обновленной информацией для базы данных;
3. На экране появляется окошко со статусом операции – успех или неудача.

Основной сценарий: «Экспорт базы данных»

Действующее лицо: Администратор

1. Администратор заходит на сайт приложения;
2. Администратор нажимает кнопку Export;
3. Файл с данными скачивается на ПК в формате json.

Дополнительный сценарий: «Просмотр информации»

Действующее лицо: Администратор

1. Администратор заходит на сайт приложения;
2. Администратор вводит поисковый запрос;
3. Администратор выбирает из списка интересующее его помещение;
4. На экране появляется карточка с информацией о помещении.

# Модель данных

## Нереляционная модель MongoDB

1. Графическое представление

Графическое представление нереляционной модели MongoDB представлено на рис. 12.

|  |
| --- |
| Рисунок 12 – Графическое представление модели MongoDB |

1. Описание назначений коллекций, типов данных и сущностей

Сущность Помещение содержит следующие свойства:

* Идентификатор – Integer
* Массив переходов – Переход

Сущность Переход содержит следующие свойства:

* Идентификатор целевого помещения – Integer
* Координаты точки отправления – {Integer, Double, Double}
* Координаты точки прибытия – {Integer, Double, Double}

Сущность Аудитория содержит следующие свойства:

* Идентификатор помещения – Integer
* Название помещения – String
* Тип помещения – Integer
* Этаж – Integer
* Информация о помещении – String
* Тег для поиска – String
* Адрес – String
* Массив стен – Стена

Сущность Стена содержит следующие свойства:

* Координаты начала – {Double, Double}
* Координаты конца – {Double, Double}

1. Оценка удельного объема информации

Пусть String в среднем занимает 100B, Integer – 4B, Double – 8B.

Стена: 32B.

Переход: 44B.

Аудитория: 412 + m × 32B, где m – количество стен.

Помещение: 4 + m × 44B, где m – количество ребер.

Общий объем одной аудитории, в котором 4 стены и 4 перехода: 720B.

Фактический объем: N × 720B, где N – количество помещений.

Чистый объем: N × 720B, где N – количество помещений.

Избыточность данной модели равна 1.

1. Запросы к модели

* Добавление аудитории:

db.room.insertOne({«id»: «1111», «name»: «Библиотека», «type»: «3», «level»: «1», «info»: «working hours: 9.00-18.00», «tags»: «библиотека, методичка, книга», «adress»: «ул. Попова, д. 5, ауд. 1111», «walls»: [{«coords\_start»:{100, 100}, «coords\_end»:{200, 200}}, {«coords\_start»:{200,200}, «coords\_end»:{200,100}}] })

db.vertex.insertOne({«id»: «1111», «edges»: [{«id\_to»:1110, «coords\_from»:{1,100, 100}, «coords\_to»:{1, 200, 200}}}]})

* Вывести все аудитории:

db.room.find({})

* Вывести граф помещений:

db.vertex.find({})

* Найти аудиторию по номеру X:

find({id: X})

## Нереляционная модель Neo4j

1. Графическое представление

Графическое представление нереляционной модели Neo4j продемонстрировано на рис. 13.

|  |
| --- |
| Рисунок 13 – Графическое представление модели Neo4j |

1. Описание назначений коллекций, типов данных и сущностей

Сущность Переход содержит следующие свойства:

* Идентификатор целевого помещения – Integer
* Идентификатор начала – Integer
* Идентификатор конца – Integer
* Вес – Double

Сущность Аудитория содержит следующие свойства:

* Идентификатор помещения – Integer
* Название помещения – String
* Тип помещения – Integer
* Этаж – Integer
* Информация о помещении – String
* Тег для поиска – String
* Адрес – String
* Массив стен – Стена

Сущность Стена содержит следующие свойства:

* Координаты начала – {Double, Double}
* Координаты конца – {Double, Double}

1. Оценка удельного объема информации

Пусть String в среднем занимает 100B, Integer – 4B, Double – 8B.

Стена: 32B.

Переход: 20B.

Аудитория: 412 + m × 32B, где m – количество стен.

Общий объем одной аудитории, в котором 4 стены и 4 перехода: 620B.

Фактический объем: N × 620B, где N – количество помещений.

Чистый объем: N × 600B, где N – количество помещений.

Избыточность данной модели равна 1.033.

1. Запросы к модели

* Добавление аудитории:

CREATE (r:Room {id: 1111, name: "Библиотека”, type: 3, level: 1, info: «working hours: 9.00-18.00», tags: «библиотека, методичка, книга», adress: «ул. Попова, д. 5, ауд. 1111», walls: [{«coords\_start»:{100, 100}, «coords\_end»:{200, 200}}, {«coords\_start»:{200,200}, «coords\_end»:{200,100}}]})

* Добавление связи между аудиторией с номерами X и Y:

MATCH (a:Room), (b:Room) WHERE a.id = X AND b.id = Y CREATE (a)-[r1:RELTYPE]->(b) CREATE (b)-[r2:RELTYPE]->(a

* Вывести все аудитории:

MATCH (r:Room) RETURN r

* Вывести все переходы:

MATCH p=()-[r:RELTYPE]->() RETURN p

* Найти аудиторию по номеру X:

MATCH (r:Room) WHERE r.id = X RETURN r

* Найти путь из аудитории с номером X в аудиторию с номером Y:

MATCH (a:Room { id: X }),(b:Room { id: Y }), p = shortestPath((a)-[\*]-(b)) RETURN p

## Реляционная модель SQL

1. Графическое представление

Графическое представление реляционной модели показано на рис. 14

|  |
| --- |
| Table  Description automatically generated  Рисунок 14 – Графическое представление модели SQL |

1. Описание назначений коллекций, типов данных и сущностей

Сущность Переход содержит следующие свойства:

* Идентификатор целевого помещения – Integer
* Координаты точки отправления – {Integer, Double, Double}
* Координаты точки прибытия – {Integer, Double, Double}

Сущность Аудитория содержит следующие свойства:

* Идентификатор помещения – Integer
* Название помещения – String
* Тип помещения – Integer
* Этаж – Integer
* Информация о помещении – String
* Тег для поиска – String
* Адрес – String
* Массив стен – Стена

Сущность Стена содержит следующие свойства:

* Идентификатор стены – Integer
* Координаты начала – {Double, Double}
* Координаты конца – {Double, Double}

Сущность Аудитория-Стена содержит следующие свойства:

* Идентификатор помещения – Integer
* Идентификатор стены – Integer

1. Оценка удельного объема информации

Пусть String в среднем занимает 100B, Integer – 4B, Double – 8B.

Стена занимает: 32B.

Переход занимает: 44B.

Аудитория занимает: 412B.

Запись в таблице связи “Аудитория – Стена”: 8B.

Общий объем одного помещения, в котором 4 стены и 4 перехода: 616B.

Фактический объем: N × 618B, где N – количество помещений.

Чистый объем: N × 594B, где N – количество помещений.

Избыточность данной модели равна 1.04.

1. Запросы к модели

Добавление аудитории:

INSERT INTO ROOM VALUES(…)

INSERT INTO EDGE VALUES(…)

for each wall:

INSERT INTO WALL VALUES(…)

INSERT INTO ROOM\_WALL VALUES(…)

Вывести все аудитории:

SELECT \* FROM ROOM

Вывести граф помещений:

SELECT \* FROM EDGE

Найти аудиторию по номеру X:

SELECT \* FROM ROOM WHERE id = X

Получить список стен в аудитории X:

SELECT \* FROM WALL WHERE id = (SELECT \* FROM ROOM\_WALL WHERE id = X)

## Сравнение моделей

Для Mongo требуется больше памяти на хранение данных, чем для SQL решения. Однако Mongo решение значительно выигрывает по количеству и сложности запросов, необходимых для работы с данными.

В сравнении с Neo4j решение на Mongo также проигрывает в объеме необходимой памяти (не считая служебную информацию). Сложность и количество запросов решений сравнимы, однако Neo4j предоставляет полезные для проекта запросы «из коробки».

В результате сравнения можно заключить, что для реализации поставленной задачи наиболее корректным было использование Neo4j в качестве системы управления базами данных.

# Разработанное приложение

## Схема экранов приложения

5.1.1. Для пользователя приложения.

|  |
| --- |
| Рисунок 15 – Схема экранов пользовательского интерфейса. Часть 1. |
| Рисунок 16 – Схема экранов пользовательского интерфейса. Часть 2. |
| Рисунок 17 – Схема экранов пользовательского интерфейса. Часть 3. |

5.1.2. Для администратора приложения.

|  |
| --- |
| Рисунок 18 – Схема экранов панели администратора |

## Использованные технологии

В процессе выполнения задачи были использованы следующие технологии:

* JavaScript;
* Node.js;
* Express.js;
* MongoDB;
* Mongoose.js;
* Pug;
* Less.

## Ссылки на приложение

Исходный код приложения доступен на GitHub по ссылке: https://github.com/moevm/nosql2h20-indoor/tree/1.0

# Выводы

## Достигнутые результаты

В ходе выполнения данной работы было разработано приложение для indoor-навигации, позволяющее просматривать информацию о помещениях, а также строить маршруты между ними. В функциональность панели администратора входят инструменты для импорта и экспорта данных. В качестве системы управления базами данных была использована MongoDB.

## Недостатки и пути для улучшения полученного решения

К недостаткам текущей реализации приложения можно отнести отсутствие учета расстояний при построении маршрута, отсутствие адаптации интерфейса для мобильных устройств, а также отсутствие возможности добавления нетекстовой информации.

## Будущее развитие решения

Дальнейшее развитие приложения предполагает учет расстояний при построении маршрута, улучшение пользовательского интерфейса, расширение возможностей по добавлению и изменению информации на странице администратора, расширение количества поддерживаемых типов информации, а также разработку мобильной версии интерфейса.

# Приложения

## Документация по сборке и развертыванию приложения

Для сборки и запуска приложения из исходного кода необходимо в консоли выполнить следующие команды:

docker-compose build

docker-compose up

После запуска приложение доступно по адресу http://0.0.0.0:3000

Панель администратора доступна по адресу http://0.0.0.0:3000/admin

## Пример файла для импорта

{"room": [ { "room\_type": 0, "tags": "1101", "\_id": 1101, "address": "ул.Попова, д.5, к.1", "floor": 1, "information": "Working day: Monday-Saturday 12.00-18.00", "name": "Dining room", "sign\_pos": [425, -550], "walls": [ { "coords\_start\_x": 300, "coords\_start\_y": -800, "coords\_end\_x": 300, "coords\_end\_y": -300 ] } ], "vertex": [ { "\_id": 1101, "transitions": [ { "target\_id": 111, "coords\_source\_floor": 1, "coords\_source\_x": 550, "coords\_source\_y": -550, "coords\_target\_floor": 1, "coords\_target\_x": 600, "coords\_target\_y": -550 } ] } ] }

# Список использованных источников

1. Документация Express.js // Express v 4.x. URL: https://expressjs.com/en/4x/api.html (дата обращения: 25.10.2020)
2. Документация Docker // Docker documentation. URL: https://docs.docker.com (дата обращения: 26.11.2020)
3. Документация MongoDB // MongoDB Documentation. URL: https://docs.mongodb.com (дата обращения: 26.12.2020)
4. Документация Mongoose.js // Mongoose v5.11.8. URL: https://mongoosejs.com/docs/ (дата обращения: 27.12.2020)