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

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: Хранение данных о проектах / участниках / ролях / интервалы контроля / цели /промахи

Студентки гр. 6381 _____ Лопатина А.С.

_____ Герасимова Д.В.

_____ Нестеркова Е.П.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2019

ЗАДАНИЕ

Студенты Лопатина А.С., Герасимова Д.В., Нестеркова Е.П.

Группа 6381

Тема проекта: Разработка приложения для хранения данных о проектах / участниках / ролях / интервалах контроля / целях /промахах

Исходные данные:
Необходимо реализовать приложение, использующее СУБД MongoDB

Содержание пояснительной записки:
«Содержание», «Введение», «Качественные требования к решению», «Сценарий использования», «Модель данных», «Разработанное приложение», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:
Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

АННОТАЦИЯ

В рамках данного курса необходимо было реализовать приложение на одну из поставленных тем. Была выбрана тема для создания приложения, которое хранит данные о проектах, участниках, их ролях и задачах в этих проектах. В приложении также должна была быть реализована функция импорта/экспорта данных.

SUMMARY

As part of this course, it was necessary to implement the application on one of the topics posed. A theme was chosen to create an application that stores data about projects, participants, their roles and tasks in these projects. The application also had to implement the function of import / export of data.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – создать приложение для хранения и отслеживания информации о проектах, их участниках, ролях и задачах.

Было решено разработать веб-приложение для просмотра списка проектов, списка сотрудников, их ролей и задач, а также дающее возможность редактировать данную информацию.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется реализовать веб-приложение, использующее СУБД MongoDB.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макеты пользовательского интерфейса

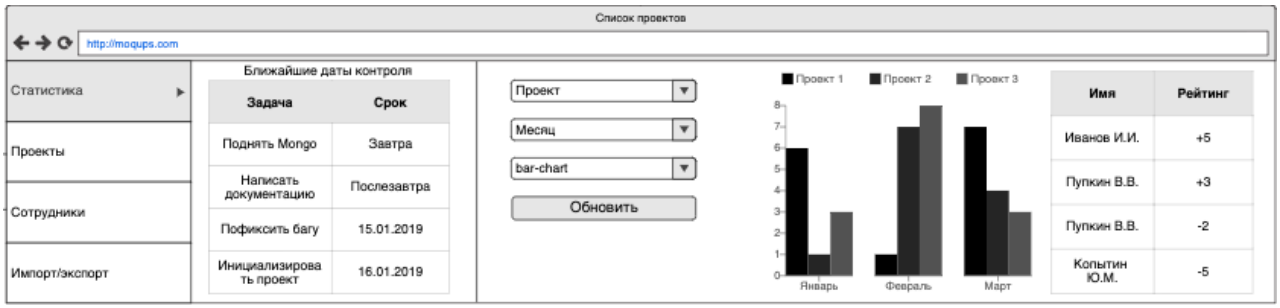

Рис. 1. Экран просмотра статистики

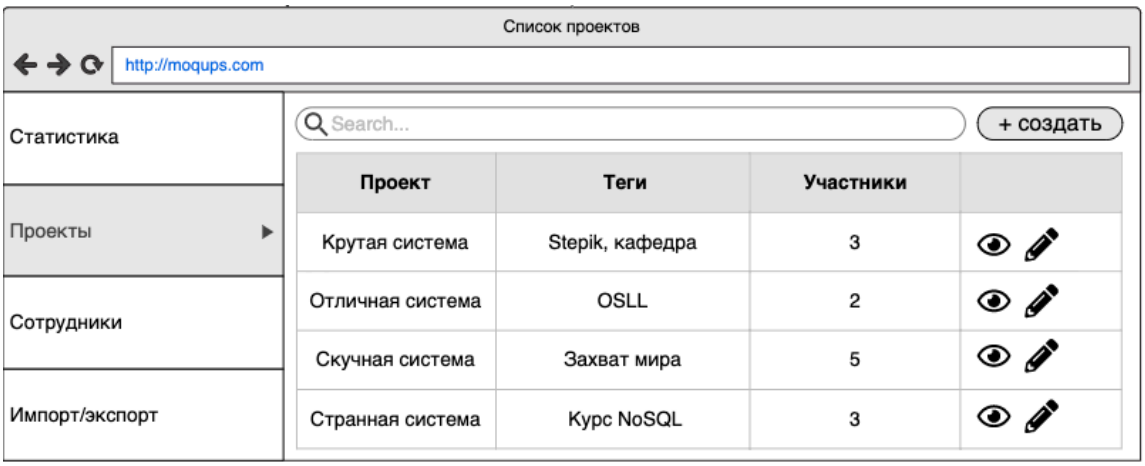

Рис. 2. Экран просмотра списка проектов

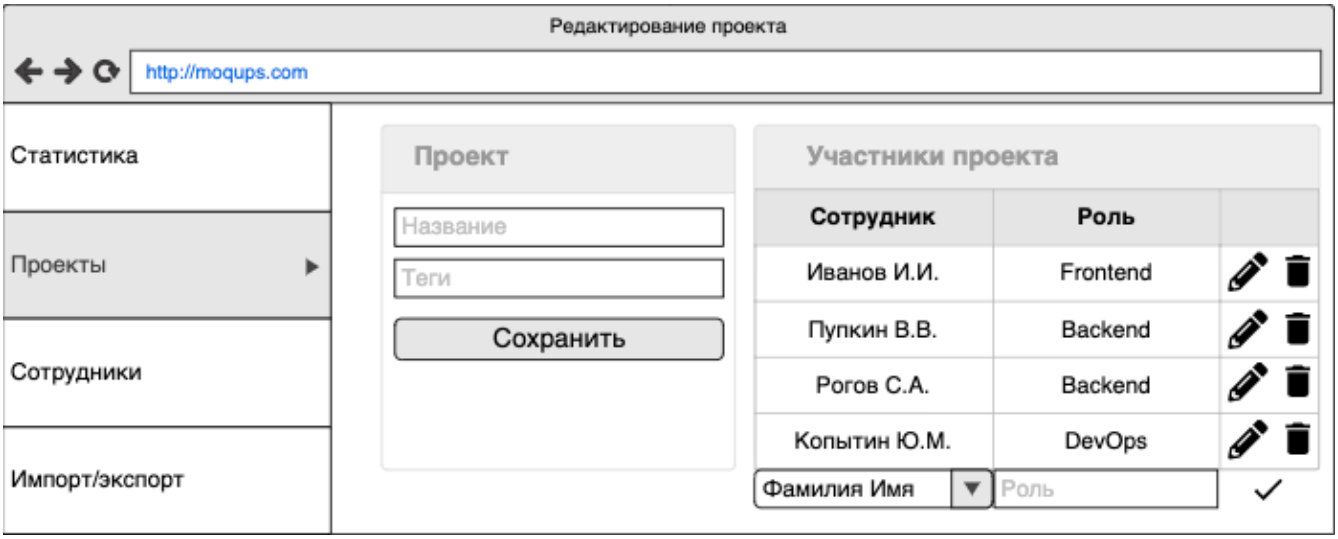

Рис. 3. Экран создания/редактирования проекта

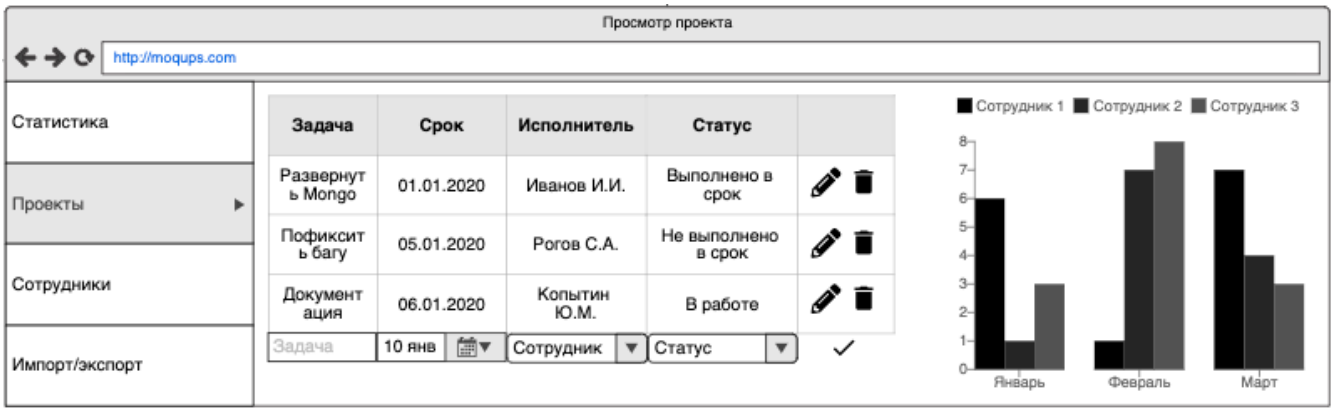

Рис. 4. Экран просмотра проекта

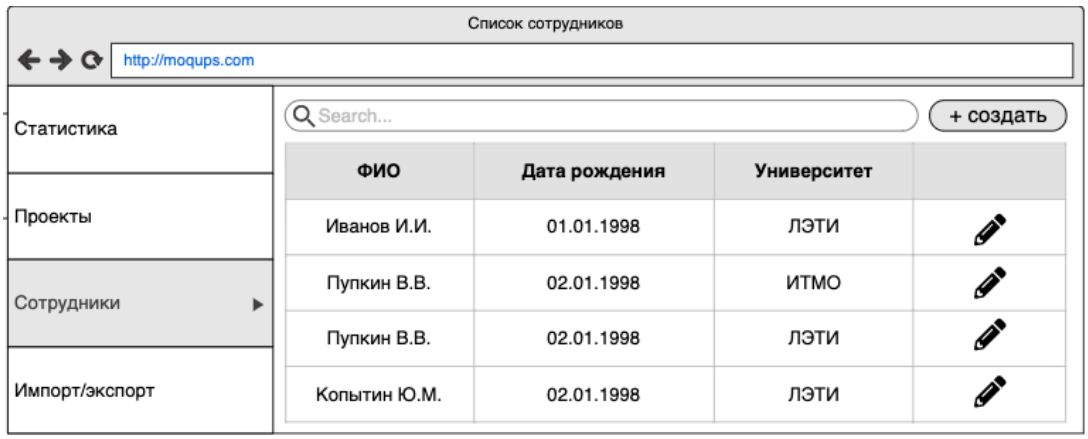

Рис. 5. Экран просмотра списка сотрудников

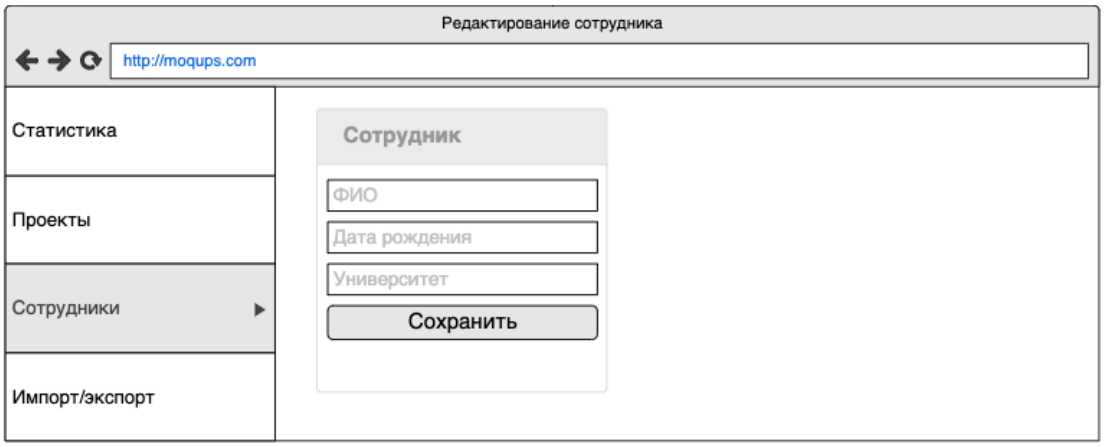

Рис. 6. Экран создания/редактирования сотрудника

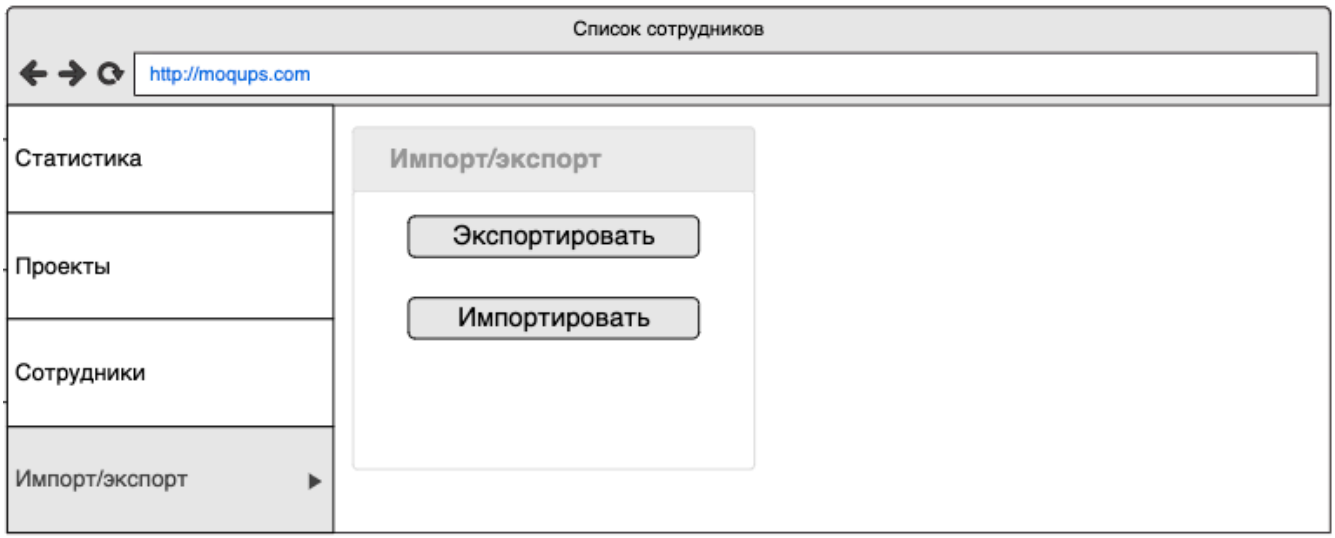

Рис. 7. Экран импорта/экспорта проекта

2.2. Описание сценариев использования

2.2.1. Сценарий использования "Просмотр списка сотрудников"

Основной сценарий:

- Пользователь открывает страницу просмотра списка проектов
- Пользователь просматривает список сотрудников

Альтернативный сценарий:

- Пользователь осуществляет поиск сотрудников с помощью ввода строки в поле поиска

2.2.2. Сценарий использования "CRUD сотрудников"

Сценарии:

1. Создание сотрудника (Create)

   Основной сценарий:

   - Пользователь открывает страницу создания сотрудника
   - Пользователь осуществляет ввода данных сотрудника в соответствующие поля
   - Пользователь нажимает кнопку "Сохранить"

   Альтернативные сценарии:

   - Сотрудник в системе уже существует
   - Пользователь отменяет создание сотрудника

2. Редактирование сотрудника (Update)

   Основной сценарий:

   - Пользователь открывает страницу редактирования сотрудника
   - Пользователь осуществляет изменение данных сотрудника в соответствующих полях
   - Пользователь нажимает кнопку "Обновить"

Альтернативные сценарии:

- Пользователь отменяет редактирование сотрудника
- Пользователь нажимает кнопку "Удалить" для удаления

сотрудника

2.2.3. Сценарий использования "CRUD проектов"

Основные и альтернативные сценарии аналогичны CRUD сотрудников, однако с добавление дополнительных сценариев:

Основной сценарий:

- Пользователь открывает страницу просмотра проекта

Альтернативные сценарии:

- Пользователь изменяет задачу проекта
- Пользователь создает задачу проекта
- Пользователь удаляет задачу проекта

2.2.4. Сценарий использования "Просмотр статистических показателей"

Основной сценарий:

- Пользователь открывает страницу просмотра статистики

Альтернативный сценарий:

- Пользователь устанавливает фильтры для просмотра нужных ему данных

2.2.5. Сценарий использования "Импорт БД"

Основной сценарий:

- Пользователь открывает страницу импорт/экспорт
- Пользователь нажимает кнопку "Импортировать"
- Пользователь выбирает файл БД в контекстном окне выбора файла

2.2.6. Сценарий использования "Экспорт БД"

Основной сценарий:

- Пользователь открывает страницу импорт/экспорт

- Пользователь нажимает кнопку "Экспортировать"
- Пользователю отправляется файл БД в окно браузера

3. МОДЕЛЬ ДАННЫХ

3.1. NoSQL модель данных

3.1.1. Графическое представление

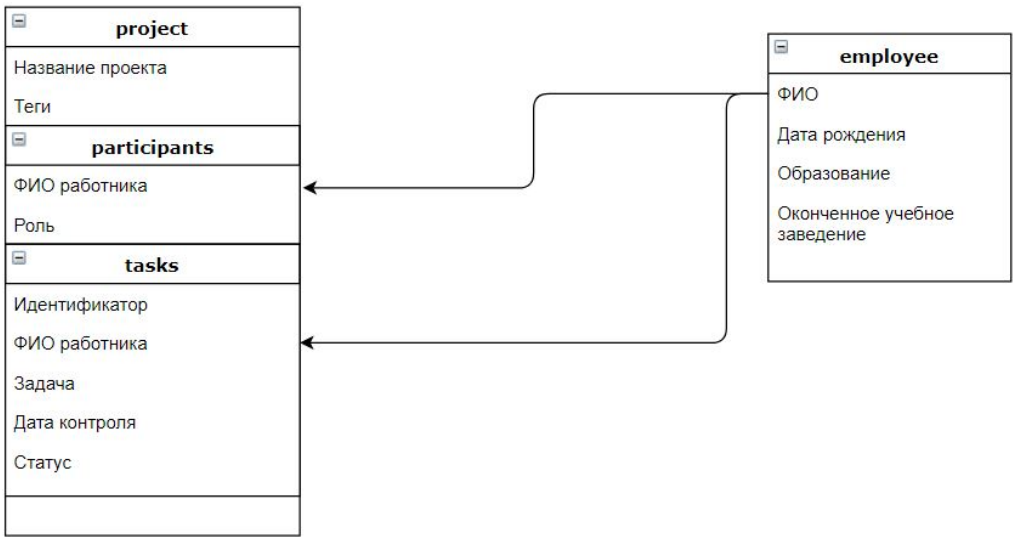

Рис. 8. Графическое представление NoSQL модели

3.1.2. Подробное описание и расчёт объема

В качестве СУБД используется MongoDB[1], в которой содержится две коллекции: employee, хранящая данные о работниках и project, содержащая сведения о проектах, его участниках, их ролях и задачах.

Каждый документ коллекции employee содержит следующие поля:

- _id - уникальный идентификатор документа. Тип - ObjectID. V = 12b
- fio - ФИО сотрудника. Тип - string. V = 2b*N, где N - средняя длина ключа задачи. На основе загруженных данных установлено, что в среднем N = 25. Тогда V = 50b
- date_of_birth - дата рождения. Тип - date. V = 8b
- education - образование. Тип - string. V = 2b*11 = 22b
- graduated_institution - оконченное учебное заведение. Тип - string. V= 2b*8 = 16b

Каждый документ коллекции project содержит следующие поля:

- _id - уникальный идентификатор документа. Тип - ObjectID. V = 12b
- name - название проекта. Тип - string. V = 2b*9 = 18b
- tags - теги. Тип - string. V = 2b*50 = 100b
- participants - участники проекта. Тип - array. Содержит массив документов с полями:
    a. employee - участник. Тип - ObjectID. V = 12b
    b. role - роль работника в данном проекте. Тип - string. V = 2b*18 = 36b
- tasks - задачи проекта. Тип - array. Содержит массив документов с полями:
    a. _id - уникальный идентификатор задачи. Тип - ObjectID. V = 12b
    b. employee - работник, которому была назначена задача. Тип ObjectID. V = 12b
    c. name - название задачи. Тип - string. V = 2b*100 = 200b
    d. date_of_control - дата контроля. Тип - date. V = 8b
    e. status - статус задачи. Тип - string. V = 2b*17 = 34b

"Чистый" объем: Объем документа в коллекции employee в среднем составляет 108b, а объем документа в коллекции project - 1124b, при условии, что в массиве participants 2 элемента, а в массиве tasks - 3. Объем всей БД V = 108b * N + 1142b * M, где N количество записей о сотрудниках в БД, M - количество записей о проектах в БД.

Фактический объем: Объем документа в коллекции employee в среднем составляет 108b, а объем документа в коллекции project - 1382b, при условии, что в массиве participants 2 элемента, а в массиве tasks - 3. Объем всей БД V = 108b * N + 1382b * M, где N количество записей о сотрудниках в БД, M - количество записей о проектах в БД.

Избыточность модели: (108b * N + 1382b * M) / (108b * N + 1124b * M)

3.1.3. Примеры запросов

- Запрос на добавление новой задачи:

```
db.getCollection("Project").update({
    _id: ObjectId("5db00ba20a1300004f00190b")
}, {
    $push: {
        tasks: {
            "employee": ObjectId("5daf5117cab8f846d8ec2223"),
            "name": "ТЕСТ",
            "date_of_control": "2019-10-18T00:00:00.000Z",
            "status": "Выполнено в срок",
            _id: new ObjectId()
        }
    }
})
```

Пример исходного кода запроса к MongoDB на добавление новой задачи в коллекцию

- Запрос на вывод задач определенного сотрудника:

```
db.Employee.find({
    fio: "Грушевская Влада Леонидовна"
}, {
    _id: 1
}) db.Project.aggregate({
    $unwind: "$tasks"
}, {
    $match: {
        "tasks.employee.$id": ObjectId("5daf537acab8f846d8ec2232")
    }
}, {
    $project: {
        "tasks._id": ObjectId("5ddec125bd6a0000b5006ffb"),
        "tasks.name": 1,
        _id: 0
    }
})
```

Пример исходного кода запроса к MongoDB на вывод задач определенного сотрудника

3.2. SQL модель данных

3.2.1. Графическое представление

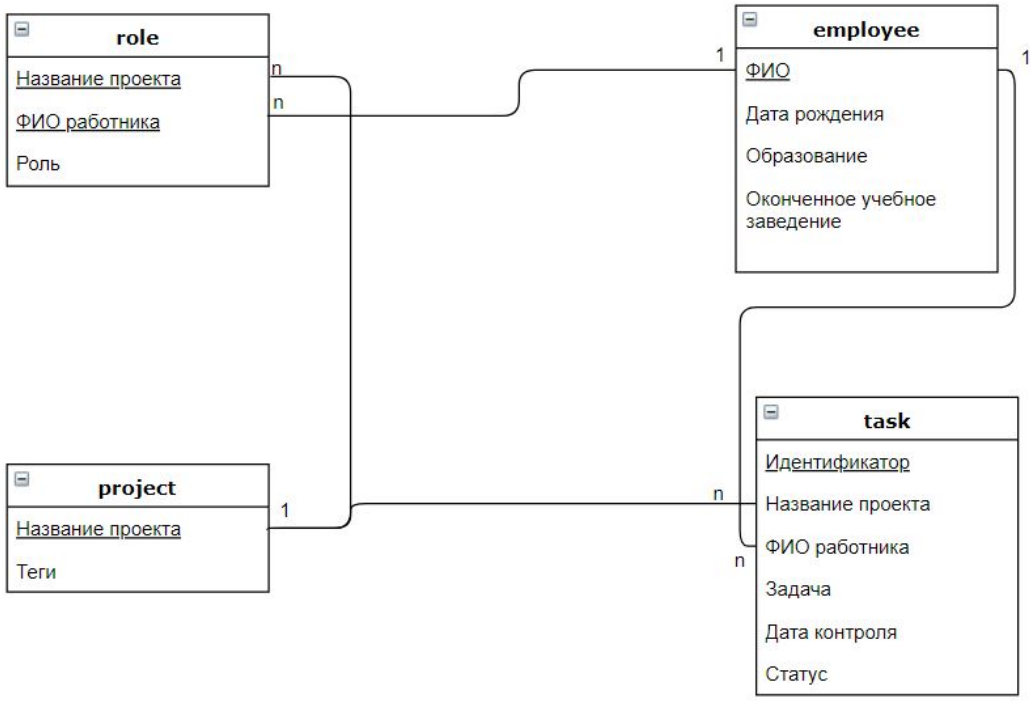

Рис. 9. Графическое представление SQL модели

3.2.2. Подробное описание и расчёт объема

В качестве реляционной СУБД использована MySQL, в которой создано 4 таблицы: employee, хранящая данные о работниках; project, содержащая сведения о проектах; role, включающая в себя данные о ролях сотрудников в проектах; task - хранящая данные о задачах.

Таблица employee содержит следующие поля:

- id - уникальный идентификатор работника. Тип - int. V = 4b
- fio - ФИО сотрудника. Тип - varchar. V = 2b*25 = 50b
- date_of_birth - дата рождения. Тип - date. V = 8b
- education - образование. Тип - varchar. V = 2b*11 = 22b
- graduated_institution - оконченное учебное заведение. Тип - varchar. V = 2b*8 = 16b

Таблица project содержит следующие поля:

- id - уникальный идентификатор проекта. Тип - int. V = 4b
- name - название проекта. Тип - varchar. V = 2b*9 = 18b

- tags - теги. Тип - varchar. V = 2b*50 = 100b

Таблица role содержит следующие поля:

- project_id - уникальный идентификатор проекта. Тип - int. V = 4b
- employee_id - уникальный идентификатор сотрудника. Тип - int. V = 4b
- role - роль работника в проекте. Тип - varchar. V = 2b*18 = 36b

Таблица task содержит следующие поля:

- id - уникальный идентификатор задачи. Тип - int. V = 4b
- project_id - уникальный идентификатор проекта. Тип - int. V = 4b
- employee_id - уникальный идентификатор сотрудника. Тип - int. V = 4b
- name - название задачи. Тип - varchar. V = 2b*100 = 200b
- date_of_control - дата контроля. Тип - date. V = 8b
- status - статус задачи. Тип - varchar. V = 2b*17 = 34b

"Чистый"объем: объем записи в таблице employee в среднем составляет 100b, в таблице project - 122b, в таблице role - 36b, в таблице task -254b. Объем всей БД V = 100b * N + 122b * M + 2 * 36b * M + 3 * 254b * M = 100b * N + 956b * M, где N количество записей о сотрудниках в БД, M - количество записей о проектах в БД.

Фактический объем: объем записи в таблице employee в среднем составляет 100b, в таблице project - 122b, в таблице role - 44b, в таблице task - 268b. Объем всей БД V = 100b * N + 122b * M + 2 * 44b * M + 3 * 268b * M = 100b * N + 1014b * M, где N количество записей о сотрудниках в БД, M - количество записей о проектах в БД.

Избыточность модели: (100b * N + 1014b * M) / (100b * N + 956b * M).

3.2.3. Примеры запросов

- Запрос на добавление новой задачи:

```
INSERT INTO task
VALUES
   ( '12745937', 3, 10, 'Интеграция переходов между слайдами блока \"Our
opportunities\" ', '2019-10-18', 'Выполнено в срок' );
```

Пример исходного кода SQL-запроса на добавление новой записи в таблицу

- Запрос на вывод задач определенного сотрудника:

```
    SELECT
          projects_db.task.id,
          projects_db.task.NAME
    FROM
          projects_db.task
          INNER JOIN projects_db.employee ON projects_db.employee.id =
projects_db.task.employee_id
    WHERE
          fio = 'Грушевская Влада Леонидовна';
```

Пример исходного кода SQL-запроса на вывод задач определенного сотрудника

3.3. Сравнение NoSQL и SQL моделей данных

В MongoDB и в SQL получается примерно одинаковый объем базы данных, так как они имеют похожую структуру. В SQL во избежание дублирования данных были созданы отдельные таблицы работников и проектов, в MongoDB проблема дублирования была решена путем создания отдельной коллекции работников, а также созданием массива участников и задач внутри коллекции проектов.

Количество запросов в Mongodb и в SQL получилось одинаковым. Когда необходимо связать информацию из разных коллекций (таблиц), в SQL используется JOIN, а в MongoDB функция $lookup. При этом время работы двух запросов в Mongodb составило 4ms, в то время как на выполнение запроса в SQL потребовалось 16ms (в базе с 5 проектами, 25 сотрудниками и 20 задачами).

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание

Для упрощения процесса разработки было принято решение разделить приложение на frontend и backend части.

Frontend-часть реализована с использованием JavaScript фреймворков React[3] и CoreUI[4]. Данные для построения графиков и таблиц, импорта и экспорта данных передаются с помощью REST API на Backend-сервер.

Backend-часть приложения реализована с использованием NodeJS фреймворка Express[5] и представляет собой набор API методов, позволяющих Frontend-приложению загружать данные из БД с помощью драйвера MongoDB NodeJS Driver [6] для отображения пользователю.

4.2. Схема экранов приложения

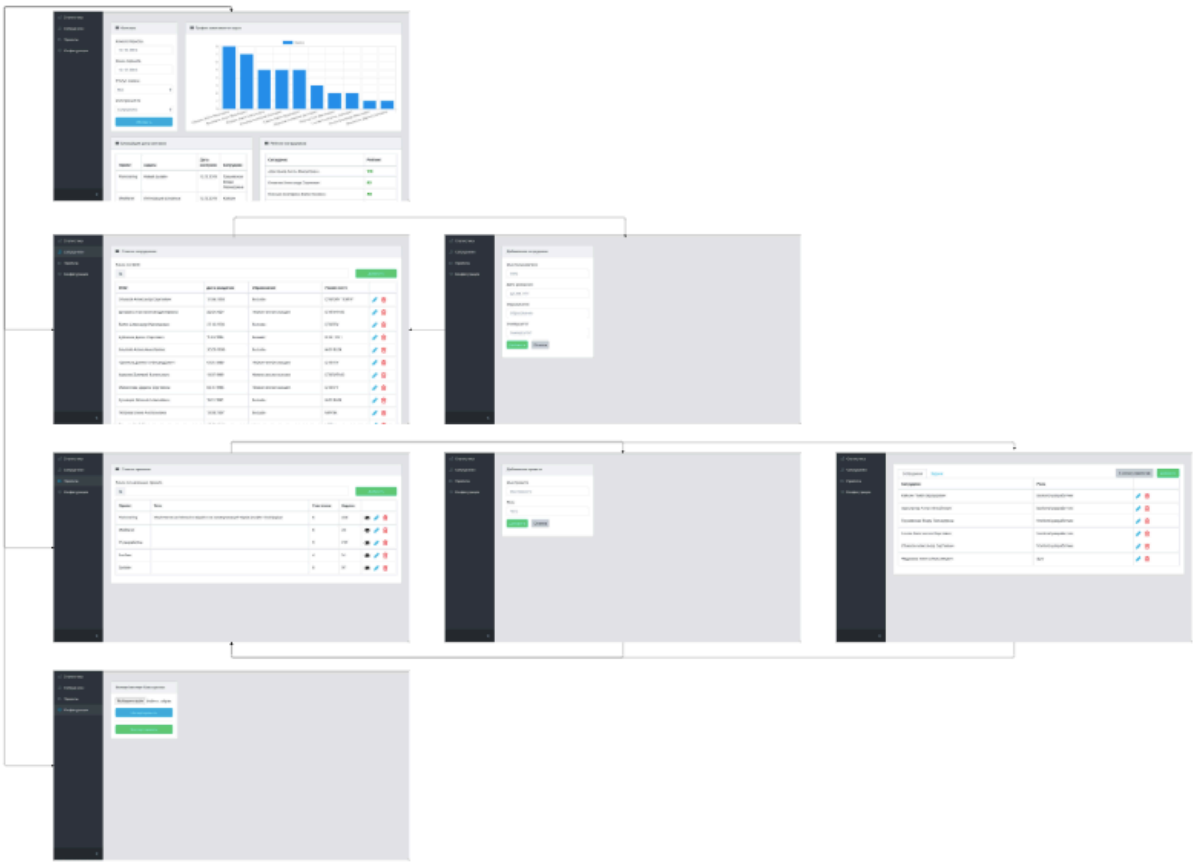

Рис. 10. Схема экранов приложения

4.3. Использованные технологии

БД: MongoDB

Backend: NodeJS

Frontend: React 16, CoreUI, Axios

4.4. Ссылка на приложение

Ссылка на приложение доступна в разделе “Список использованных источников” [7]

ЗАКЛЮЧЕНИЕ

В ходе работы было реализовано приложение, в котором можно просмотреть информацию о проектах и их участниках, об их ролях в данных проектах, а также задачах. В приложении есть функции импорта/экспорта данных. Кроме того, пользователь может просмотреть различную статистику, выбирая необходимые фильтры. Таким образом, цель, поставленная перед началом работы, достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация MongoDB: https://docs.mongodb.com/manual/
2. Документация MySQL: https://dev.mysql.com/doc/
3. Документация React 16: https://reactjs.org/docs/getting-started.html
4. Документация CoreUI: https://coreui.io/docs/getting-started/introduction/
5. Документация NodeJS: https://nodejs.org/ru/docs/
6. Документация MongoDB NodeJS Driver: https://mongodb.github.io/node-mongodb-native/
7. Исходный код приложения: https://github.com/moevm/nosql2h19-projects