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: ИС Почты

Студент гр. 9303 _____ Ахримов А.М.

Студент гр. 9303 _____ Муратов Р.А.

Студент гр. 9303 _____ Низовцов Р.С.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2022

ЗАДАНИЕ

Студенты

Ахримов А.М.

Муратов Р.А.

Низовцов Р.С.

Группы 9303

Тема проекта: Разработка информационной системы почты

Исходные данные:

Необходимо реализовать клиентскую и серверную части для информационной системы почты с использованием документоориентированной СУБД MongoDB.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Вывод»

«Приложения»

«Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 08.09.2022

Дата сдачи реферата: 17.12.2022

Дата защиты реферата: 17.12.2022

Студент гр. 9303	_____	Ахримов А.М.
Студент гр. 9303	_____	Муратов Р.А.
Студент гр. 9303	_____	Низовцов Р.С.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В рамках курса была реализована информационная система для работы почты, в которой моделируется отправка писем, бандеролей и посылок. Разработаны клиентский (React) и серверный (Golang) программные модули. Для получения практического опыта работы с нереляционными СУБД было выбрано MongoDB - решение с документоориентированной моделью данных. Основной принцип при разработке системы – переложить как можно больше работы по обработке данных «на плечи» СУБД. Для развертывания системы использовался Docker.

Программный код и инструкции по разворачиванию системы можно найти в репозитории GitHub.

SUMMARY

As part of the course, an information system was implemented for the work of the post, in which the sending of letters and parcels is simulated. Client (React) and server (Golang) software modules have been developed. To gain practical experience with non-relational DBMS, MongoDB was chosen - a solution with a document-oriented data model. The main principle in the development of the system is to shift as much data processing work as possible “on the shoulders” of the DBMS. Docker was used to deploy the system.

Program code and instructions for deploying the system can be found in the GitHub repository.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – создать решение для хранения и отображения информации о посылках и сотрудниках почтовых отделении.

Было реализовано клиент-серверное приложение с веб-интерфейсом, которое позволяет:

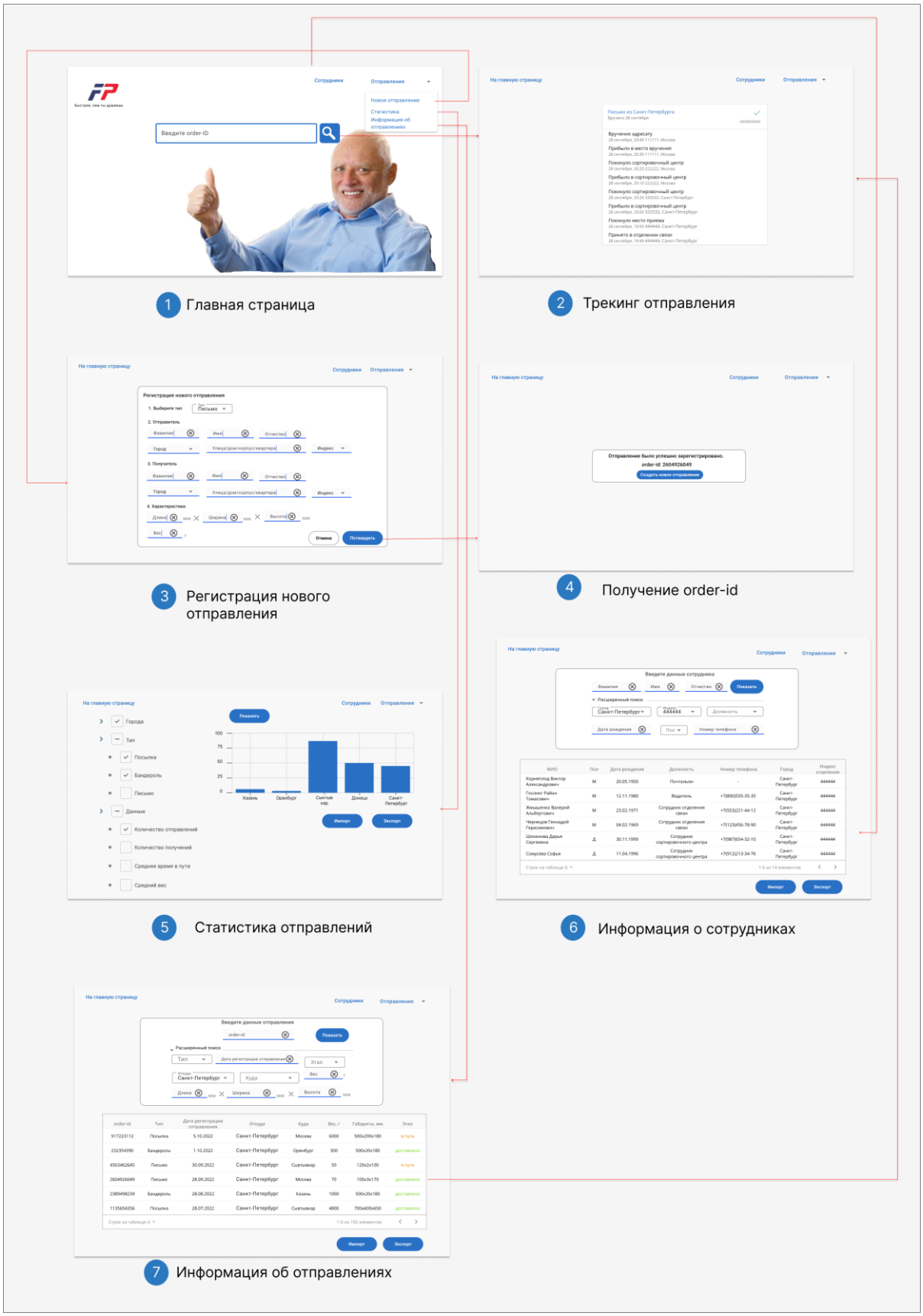
1. Получать информацию о посылке по ее идентификатору, который выдается при ее оформлении
2. Просматривать сотрудников почтовых отделении с возможностью фильтрации поисковой выдачи
3. Искать посылки и фильтровать их по определенным полям (размер, вес, пункт отправления, пункт назначения и т.д.)
4. Получать статистику отправлений по следующим параметрам: время доставки, тип отправления и пункт назначения (отправки).

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать клиент-серверное приложение с веб-интерфейсом, использующем в качестве хранилища данных БД с нереляционной моделью данных (MongoDB).

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет UI



2.2. Описание сценариев

Сценарий использование – «Просмотр состояния отправления»

Пользователь*:* Сотрудник почты.

Основной сценарий:

1. Пользователь на главной странице (1) вводит в поисковое поле order-id.
2. Пользователь нажимает на кнопку со значком лупы.
3. Пользователю показывается страница с текущим состоянием отправления (2).

Альтернативный сценарий:

- Пользователь вводит несуществующий order-id, система сообщает ему об этом модальным окном.
- Пользователь находит отправление через таблицу отправлений:
    - Пользователь на главной странице (1) нажимает на Отправления→Информация об отправлениях.
    - Пользователь на странице вводит order-id и/или пользуется расширенным поиском: город отправителя, город получателя, тип отправления, дата регистрации отправления, этап доставки отправления, вес и габариты.
    - Пользователь нажимает на кнопку “Показать”.
    - Пользователь находит в таблице отправление и нажимает на него.
    - Пользователю показывается страница с текущим состоянием отправления (2).

Сценарий использование - “Регистрация нового отправления”

Пользователь: Сотрудник почты.

Основной сценарий:

1. Пользователь на главной странице (1) нажимает на Отправления→Новое отправление.
2. Пользователь вводит данные об отправлении (3): тип, данные отправителя, данные получателя, характеристики.

3. Пользователь нажимает на кнопку “Подтвердить”.
4. Пользователя информируют об успешной регистрации, показывается order-id его отправления (4) и предлагается создать новое отправление.

Альтернативный сценарий:

Пользователь вводит недостаточное количество информации и нажимает на кнопку “Подтвердить”, система сообщает ему об этом модальным окном.

Сценарий использование - “Просмотр статистики по отправлениям”

Пользователь: Сотрудник почты.

Основной сценарий:

1. Пользователь на главной странице (1) нажимает на Отправления→Статистика.
2. Пользователь на странице статистике (5) выбирает город, тип и данные.
3. Пользователь нажимает на кнопку “Показать”.
4. Система показывает пользователю соответствующую гистограмму.

Сценарий использование - “Просмотр сотрудников”

Пользователь: Сотрудник почты.

Основной сценарий:

1. Пользователь на главной странице (1) нажимает на Сотрудники.
2. Пользователь на странице с информацией о сотрудниках (6) вводит ФИО сотрудника и/или пользуется расширенным поиском: город, индекс отделения, должность, дата рождения, пол, номер телефона.
3. Пользователь нажимает на кнопку “Показать”.
4. Система показывает таблицу с данными о сотрудниках.

Пользователь имеет возможность добавлять отправления, регистрируя их, и просматривать данные с помощью веб-интерфейса.

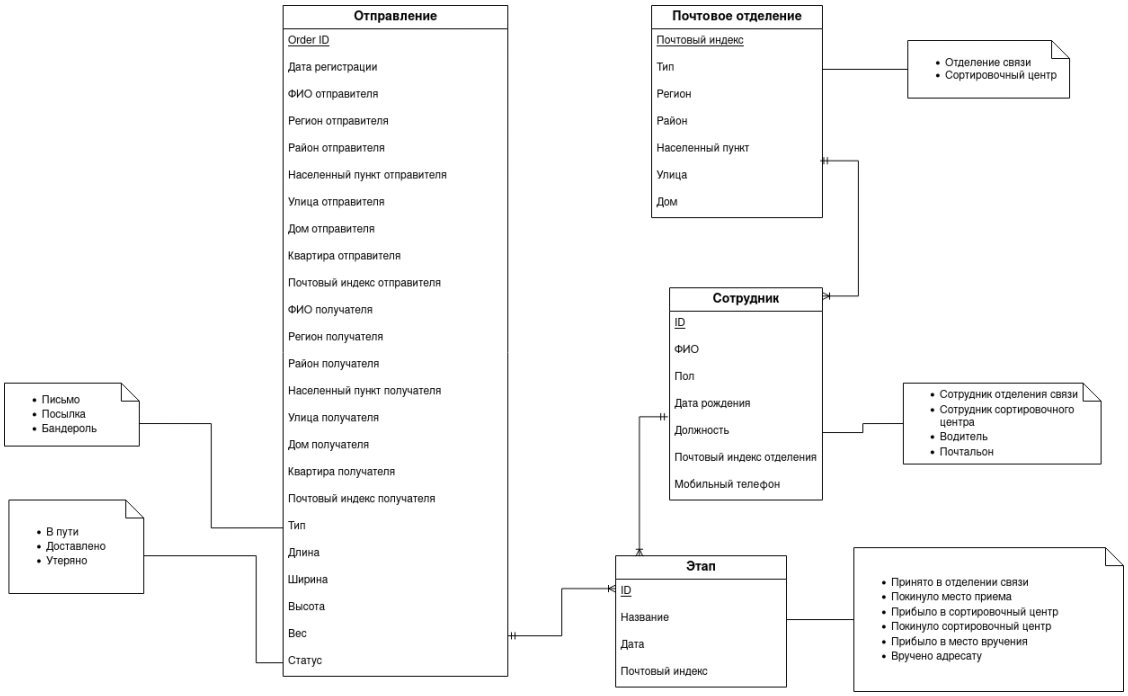
Импорт и экспорт отправлений осуществляется с помощью соответствующих элементов интерфейса: кнопок «Импорт» и «Экспорт» на

странице с отправлениями (7). Для импорта и экспорта используются файла формата JSON.

Для текущего решения в равной степени будут производится операции чтения и записи, т.к. с одной стороны, добавляются новые отправления и их состояние обновляется, с другой – просматриваются состояния отправлении и анализируется статистика по отправлениям.

3. МОДЕЛЬ ДАННЫХ

3.1. ER диаграмма



3.2. Нереляционная СУБД

Sending

```
{
  "_id": <ObjectId>,
  "order_id": "123e4567-e89b-12d3-a456-426655440000",
  "registration_date": ISODate("2000-01-1T00:00:00Z"),
  "sender": {
    "name": "Налсур",
    "surname": "Мулюков",
    "middle_name": "Рустэмович",
    "address": {
      "postcode": "453870",
      "region": "Республика Башкортостан",
      "district": "Мелеузовский район",
      "settlement": "пос. Нутуш",
      "street": "ул. Ленина",
      "building": "42"
    }
  },
  "receiver": {
    "name": "Екатерина",
    "surname": "Фемининсткова",
    "address": {
      "postcode": "123456",
      "settlement": "г. Москва",
      "street": "ул. Мира",
      "building": "1",
      "apartment": "1"
    }
  },
  "type": "Посылка",
  "size": {
    "length": 100,
    "width": 73,
    "height": 42
  },
  "weight": 1000,
  "stages": [
    {
      "name": "Принято в отделении связи",
      "timestamp": ISODate("2000-01-1T00:00:00Z"),
      "postcode": "453870",
      "employee_id": <ObjectId>
    },
      ...,
    {
      "name": "Вручено адресату",
      "timestamp": ISODate("2000-02-1T00:00:00Z"),
      "postcode": "123456",
      "employee_id": <ObjectId>
    }
  ],
  "status": "Доставлено"
}
```

Post office

```
{
  "_id": <ObjectId>
  "type": "Отделение связи",
  "address": {
    "postcode": "453250",
    "region": "Республика Башкортостан",
    "district": "Стерлитамакский район",
    "settlement": "г. Салават",
    "street": "бул. Салавата Юлаева",
    "building": "18А"
  },
  "employees": [
      <ObjectId>,
      ...,
      <ObjectId>
  ]
}
```

Employee

```
{
  "_id": <ObjectId>
  "surname": "Гослинг",
  "name": "Райан",
  "middle_name": "Томасович",
  "gender": "М",
  "birth_date": ISODate("1980-11-12T00:00:00Z"),
  "position": "Водитель",
  "phone_number": "88005553535"
}
```

Сущности

Адрес (660 байт):

- postcode: string - почтовый индекс (6 байт)
- region: string - регион (субъект) (150 байт)
- district: string - район (150 байт)
- settlement: string - населенный пункт (150 байт)
- street: string - улица (150 байт)
- building: string - дом (50 байт)
- apartment: string - квартира (4 байт)

Клиент (отправитель или получатель) (960 байт):

- name: string - имя (100 байт)
- surname: string - фамилия (100 байт)
- middle_name: string - отчество (100 байт)
- address: object - адрес (660 байт)

Размер (24 байта):

- length: int - длина (8 байт)
- width: int - ширина (8 байт)
- height: int - высота (8 байт)

Этап (136 байта):

- name: string - название этапа (110 байт). Допустимые значения:
    - Принято в отделении связи
    - Покинуло место приема
    - Прибыло в сортировочный центр
    - Покинуло сортировочный центр
    - Прибыло в место вручения
    - Вручено адресату
- timestamp: date - время в формате YYYY-MM-DDThh:mm:ssZ (8 байт)
- postcode: string - почтовый индекс (6 байт)

- employee_id: ObjectId - идентификатор сотрудника, который обработал заказ на этом этапе (12 байт)

Отправление ($2012 + 136 * St$ байт):

- _id: ObjectId - идентификатор (12 байт)
- registration_date: date - дата регистрации посылки в формате YYYY-MM-DDThh:mm:ssZ (8 байт)
- sender: object - отправитель (960 байт)
- receiver: object - получатель (960 байт)
- type: string - тип посылки (20 байт). Допустимые значения:
  - Письмо
  - Посылка
  - Бандероль
- size: object - размер (24 байта)
- weight: int - вес (8 байт)
- stages: object[] - пройденные этапы ($136 * St$ байт)
- status: string - текущий статус (20 байт). Допустимые значения:
  - В пути
  - Доставлено
  - Утеряно

Работник (452 байт):

- _id: ObjectId - идентификатор (12 байт)
- name: string - имя (100 байт)
- surname: string - фамилия (100 байт)
- middle_name: string - отчество (100 байт)
- gender: string - пол (1 байт). Допустимые значения:
  - М
  - Ж
- birth_date: date - дата рождения в формате YYYY-MM-DDThh:mm:ssZ (8 байт)

- position: string - должность (120 байт). Допустимые значения:
    - Сотрудник отделения связи
    - Сотрудник сортировочного центра
    - Водитель
    - Почтальон
- phone_number: string - номер телефона в формате 8XXXXXXXXXX (11 цифр) (11 байт)

Почтовое отделение ($747 + 12 * E$ байт):

- _id: ObjectId - идентификатор поля (12 байт)
- type: string - тип отделения (75 байт). Допустимые значения:
    - Отделение связи
    - Сортировочный центр
- address: object - адрес (660 байт)
- employees: ObjectId[] - работники отделения ($12 * E$ байт)

Коллекции

Почтовое отделение (PostOffice) - в качестве документов используются сущности “Почтовое отделение”

Отправление (Sending) - в качестве документов используются сущности “Отправление”

Сотрудник (Employee) - в качестве документов используются сущности “Работник”

Оценка удельного объёма информации

Пусть S - количество отправлений, E - среднее количество работников, P - количество почтовых отделений, St - среднее количество пройденных этапов на одно отправление.

С определенной долей условности можно считать, что E=200, P=50, St=10.

Документ в коллекции PostOffice:

- $735 + 12 \cdot E$ байт (“чистый”)

- $747 + 12 \cdot E$ байт (фактический)

Документ в коллекции Sending:

- $2000 + 124 \cdot St$ байт (“чистый”)
- $2012 + 136 \cdot St$ байт (фактический)

Документ в коллекции Employee:

- 440 байт (“чистый”)
- 452 байт (фактический)

“Чистый” объём данных

Таким образом, “чистый” объём данных в байтах вычисляется по следующей формуле:

$$data_{net} = (735 \cdot P + 12 \cdot E) + S \cdot (2000 + 124 \cdot St) + 440 \cdot E$$

Или:

$$data_{net} = 127150 + 3240 \cdot S$$

Фактический объём данных

Фактический объём данных в байтах вычисляется по формуле ниже:

$$data_{gross} = (747 \cdot P + 12 \cdot E) + S \cdot (2012 + 136 \cdot St) + 452 \cdot E$$

Или:

$$data_{gross} = 130150 + 3372 \cdot S$$

Избытычность

Избытычность модели данных можно вычислить по этой формуле:

$$\frac{data_{gross}}{data_{net}} = \frac{130150 + 3372 \cdot S}{127150 + 3240 \cdot S} = 1 + \frac{3000 + 132 \cdot S}{127150 + 3240 \cdot S}$$

Направление роста модели при увеличении количества объектов каждой сущности

При увеличении количества отправлений S, среднего количества работников в одном почтовом отделении E, количества почтовых отделений P или среднего количества пройденных этапов на одно отправление St рост модели будет линейным.

Запросы к БД для реализации сценариев использования

“Просмотр состояния отправления”

Запрос (основной сценарий):

```
db.sending.aggregate([
  {
    $match: {
      order_id: "123e4567-e89b-12d3-a456-426655440000"
    }
  },
  {
    $unwind: "$stages"
  },
  {
    $replaceWith: "$stages"
  },
  {
    $project: {
      employee_id: 0
    }
  }
])
```

Ответ:

```
[
  {
    "name": "Принято в отделении связи",
    "postcode": "123456",
    "timestamp": ISODate("2000-01-01T00:00:00Z")
  },
  {
    "name": "Покинуло место приема",
    "postcode": "123456",
    "timestamp": ISODate("2000-01-02T00:00:00Z")
  },
  {
    "name": "Прибыло в сортировочный центр",
    "postcode": "111222",
    "timestamp": ISODate("2000-01-03T00:00:00Z")
  }
]
```

Количество запросов: 1

Количество задействованных коллекций: 1

Запрос (альтернативный сценарий, при поиске заданы все параметры):

```
db.sending.find({
  order_id: "123e4567-e89b-12d3-a456-426655440000",
  type: "Посылка",
  registration_date: ISODate("2000-01-01T00:00:00Z"),
  "sender.address.settlement": "г. Мелеуз",
  "receiver.address.settlement": "г. Ишимбай",
  weight: 10,
  size: {
    length: 1,
    width: 2,
    height: 3
```

```
    },
    status: "В пути"
  },
  {
    _id: 0,
    "stages.employee_id": 0
  })
```

Ответ:

```
[
  {
    "order_id": "123e4567-e89b-12d3-a456-426655440000",
    "receiver": {
      "address": {
        "building": "4",
        "district": "Ишимбайский район",
        "postcode": "654321",
        "region": "Респ. Башкортостан",
        "settlement": "г. Ишимбай",
        "street": "ул. Октябрьская"
      },
      "name": "Айнур",
      "surname": "Ибрагимов"
    },
    "registration_date": ISODate("2000-01-01T00:00:00Z"),
    "sender": {
      "address": {
        "building": "12",
        "district": "Салаватский район",
        "postcode": "123456",
        "region": "Респ. Башкортостан",
        "settlement": "г. Мелеуз",
        "street": "площадь Ленина"
      },
      "middle_name": "Азатович",
      "name": "Руслан",
      "surname": "Муратов"
    },
    "size": {
      "height": 3,
      "length": 1,
      "width": 2
    },
    "stages": [
      {
        "name": "Принято в отделении связи",
        "postcode": "123456",
        "timestamp": ISODate("2000-01-01T00:00:00Z")
      },
      {
        "name": "Покинуло место приема",
        "postcode": "123456",
        "timestamp": ISODate("2000-01-02T00:00:00Z")
      },
      {
        "name": "Прибыло в сортировочный центр",
        "postcode": "111222",
        "timestamp": ISODate("2000-01-03T00:00:00Z")
```

```
      }
    ],
    "status": "В пути",
    "type": "Посылка",
    "weight": 10
  }
]
```

Количество запросов: 1

Количество задействованных коллекций: 1

“Регистрация нового отправления”

Запрос:

```
db.sending.insertOne({
  order_id: "123e4567-e89b-12d3-a456-426655441234",
  type: "Бандероль",
  registration_date: ISODate("2022-10-27T00:00:00Z"),
  sender: {
    name: "Эмиль",
    surname: "Газизов",
    middle_name: "Наилевич",
    address: {
      postcode: "436123",
      region: "Респ. Татарстан",
      district: "Альметьевский район",
      settlement: "г. Альметьевск",
      street: "ул. Советская",
      building: "147"
    }
  },
  receiver: {
    name: "Регина",
    surname: "Мулюкова",
    middle_name: "Радиковна",
    address: {
      postcode: "194505",
      settlement: "г. Москва",
      street: "ул. Победы",
      building: "9"
    }
  },
  weight: 1000,
  size: {
    length: 20,
    width: 10,
    height: 5
  },
  status: "В пути",
  stages: [
    {
      employee_id: "e_4"
      name: "Принято в отделении связи",
      postcode: "123456",
      timestamp: ISODate("2022-10-27T00:00:00Z")
    }
  ]
```

```
})
```

Количество запросов: 1

Количество задействованных коллекций: 1

“Просмотр статистики по отправлениям”

Запрос (количество отправлений):

```
db.sending.aggregate([
  {
    "$match": {
      "sender.address.settlement": {
        "$in": [
          "г. Уфа",
          "г. Салават"
        ]
      },
      "type": {
        "$in": [
          "Письмо",
          "Посылка"
        ]
      }
    }
  },
  {
    "$group": {
      "_id": "$sender.address.settlement",
      "shipments_number": {
        $sum: 1
      }
    }
  },
  {
    "$project": {
      _id: 0,
      settlement: "$_id",
      shipments_number: "$shipments_number"
    }
  }
])
```

Ответ (количество отправлений):

```
[
  {
    "settlement": "г. Салават",
    "shipments_number": 2
  },
  {
    "settlement": "г. Уфа",
    "shipments_number": 1
  }
]
```

Примечание: запрос для “количество получений” аналогичен приведенному выше, за исключением того, что используется $receiver.address.settlement

Количество запросов: 1

Количество задействованных коллекций: 1

Запрос (среднее время в пути с фильтрацией по городам и типам):

```
db.sending.aggregate([
  {
    "$match": {
      "sender.address.settlement": {
        "$in": [
          "г. Мелеуз",
          "г. Салават"
        ]
      },
      "type": {
        "$in": [
          "Письмо",
          "Посылка"
        ]
      },
      "status": "Доставлено"
    }
  },
  {
    "$group": {
      "_id": {
        settlement: "$sender.address.settlement"
      },
      "average_time": {
        $avg: {
          "$dateDiff": {
            startDate: "$registration_date",
            endDate: {
              "$getField": {
                field: "timestamp",
                input: {
                  $last: "$stages"
                }
              }
            },
            unit: "second",
            timezone: "GMT",
            startOfWeek: "mon"
          }
        }
      }
    }
  },
  {
    "$project": {
      _id: 0,
```

```
      settlement: "$_id.settlement",
      average_time: "$average_time"
    }
  }
])
```

Ответ (среднее время в пути):

```
[
  {
    "average_time": 172800,
    "settlement": "г. Салават"
  },
  {
    "average_time": 600,
    "settlement": "г. Мелеуз"
  }
]
```

Количество запросов: 1

Количество задействованных коллекций: 1

Запрос (средний вес типов отправлений с фильтрацией по городам и типам):

```
db.sending.aggregate([
  {
    "$match": {
      "sender.address.settlement": {
        "$in": [
          "г. Мелеуз",
          "г. Салават"
        ]
      },
      "type": {
        "$in": [
          "Письмо",
          "Посылка"
        ]
      }
    }
  },
  {
    "$group": {
      "_id": {
        type: "$type"
      },
      "average_weight": {
        $avg: "$weight"
      }
    }
  },
  {
    "$project": {
      _id: 0,
      type: "$_id.type",
      average_weight: "$average_weight"
    }
```

```
  }
])
```

Ответ (средний вес):

```
[
  {
    "average_weight": 550,
    "type": "Бандероль"
  },
  {
    "average_weight": 100,
    "type": "Посылка"
  },
  {
    "average_weight": 1,
    "type": "Письмо"
  }
]
```

Количество запросов: 1

Количество задействованных коллекций: 1

“Просмотр сотрудников”

Запрос (при поиске заданы все параметры):

```
db.post_office.aggregate([
  {
    $match: {
      "address.postcode": "123456"
    }
  },
  {
    $unwind: {
      path: "$employees"
    }
  },
  {
    $lookup: {
      from: "employee",
      localField: "employees",
      foreignField: "_id",
      as: "employee_data"
    }
  },
  {
    $unwind: {
      path: "$employee_data"
    }
  },
  {
    $replaceWith: "$employee_data"
  },
  {
    $match: {
      name: "Райан",
      surname: "Гослинг",
      middle_name: "Томасович",
```

```
        gender: "M",
        birth_date: ISODate("1980-11-12T00:00:00Z"),
        position: "Водитель",
        phone_number: "88005553535"
      }
    },
    {
      $project: {
        _id: 0
      }
    }
])
```

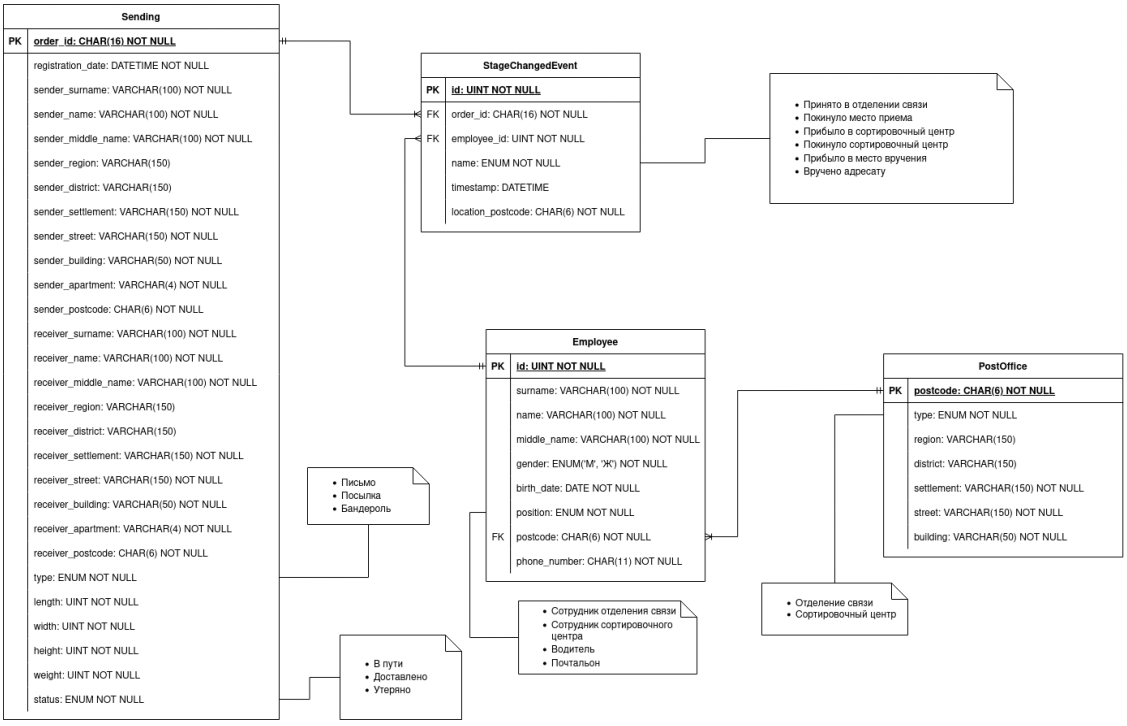
Ответ:

```
[
  {
    "birth_date": ISODate("1980-11-12T00:00:00Z"),
    "gender": "M",
    "middle_name": "Томасович",
    "name": "Райан",
    "phone_number": "88005553535",
    "position": "Водитель",
    "surname": "Гослинг"
  }
]
```

Количество запросов: 1

Количество задействованных коллекций: 2

3.3. Реляционная СУБД



Сущности

Отправление (Sending) (2012 байт):

- order_id: CHAR(16) - идентификатор в формате UUID (16 байт)
- registration_date: DATETIME - дата регистрации посылки в формате YYYY-MM-DDThh:mm:ssZ (8 байт)
- sender_surname: VARCHAR(100) - фамилия отправителя (100 байт)
- sender_name: VARCHAR(100) - имя отправителя (100 байт)
- sender_middle_name: VARCHAR(100) - отчество отправителя (100 байт)
- sender_region: VARCHAR(150) - регион (субъект) отправителя (150 байт)
- sender_district: VARCHAR(150) - район отправителя (150 байт)
- sender_settlement: VARCHAR(150) - населенный пункт отправителя (150 байт)
- sender_street: VARCHAR(150) - улица отправителя (150 байт)
- sender_building: VARCHAR(50) - дом отправителя (50 байт)
- sender_apartment: VARCHAR(4) - квартира отправителя (4 байт)
- sender_postcode: CHAR(6) - почтовый индекс отправителя (4 байт)
- receiver_surname: VARCHAR(100) - фамилия получателя (100 байт)
- receiver_name: VARCHAR(100) - имя получателя (100 байт)
- receiver_middle_name: VARCHAR(100) - отчество получателя (100 байт)
- receiver_region: VARCHAR(150) - регион (субъект) получателя (150 байт)
- receiver_district: VARCHAR(150) - район получателя (150 байт)
- receiver_settlement: VARCHAR(150) - населенный пункт получателя (150 байт)
- receiver_street: VARCHAR(150) - улица получателя (150 байт)
- receiver_building: VARCHAR(50) - дом получателя (50 байт)

- receiver_apartment: VARCHAR(4) - квартира получателя (4 байт)
- receiver_postcode: CHAR(6) - почтовый индекс получателя (4 байт)
- type: ENUM - тип посылки (20 байт). Допустимые значения:
    - Письмо
    - Посылка
    - Бандероль
- length: UINT - длина (8 байт)
- width: UINT - ширина (8 байт)
- height: UINT - высота (8 байт)
- weight: UINT - вес (8 байт)
- status: ENUM - текущий статус (20 байт). Допустимые значения:
    - В пути
    - Доставлено
    - Утеряно

Работник (Employee) (454 байт):

- id: UINT - идентификатор (8 байт)
- name: VARCHAR(100) - имя (100 байт)
- surname: VARCHAR(100) - фамилия (100 байт)
- middle_name: VARCHAR(100) - отчество (100 байт)
- gender: ENUM - пол (1 байт). Допустимые значения:
    - М
    - Ж
- birth_date: DATE - дата рождения в формате YYYY-MM-DD (8 байт)
- position: ENUM - должность (120 байт). Допустимые значения:
    - Сотрудник отделения связи
    - Сотрудник сортировочного центра
    - Водитель
    - Почтальон

- phone_number: CHAR(11) - номер телефона в формате 8XXXXXXXXXX (11 цифр) (11 байт)
- postcode: CHAR(6) - почтовый индекс отделения (место работы) (6 байт)

Почтовое отделение (PostOffice) (731 байт):

- postcode: CHAR(6) - почтовый индекс отделения (6 байт)
- type: ENUM - тип отделения (75 байт). Допустимые значения:
  - Отделение связи
  - Сортировочный центр
- region: VARCHAR(150) - регион (субъект) (150 байт)
- district: VARCHAR(150) - район (150 байт)
- settlement: VARCHAR(150) - населенный пункт (150 байт)
- street: VARCHAR(150) - улица (150 байт)
- building: VARCHAR(50) - дом (50 байт)

Этап (StageChangedEvent) (156 байта):

- id: UINT - идентификатор (8 байт)
- order_id: CHAR(16) - идентификатор отправления, к которому относится этап, в формате UUID (16 байт)
- employee_id: UINT - идентификатор сотрудника, который обработал отправление на этом этапе (8 байт)
- name: ENUM - название этапа (110 байт). Допустимые значения:
  - Принято в отделении связи
  - Покинуло место приема
  - Прибыло в сортировочный центр
  - Покинуло сортировочный центр
  - Прибыло в место вручения
  - Вручено адресату
- timestamp: DATETIME - время в формате YYYY-MM-DDThh:mm:ssZ (8 байт)

- postcode: CHAR(6) - почтовый индекс (6 байт)

Оценка удельного объёма информации

Строка в таблице Sending:

- 2012 байт (“чистый”)
- 2012 байт (фактический)

Пояснение: “чистый” и фактический объемы совпадают, т.к. в строках таблицы Sending хранятся только domain (business) данные.

Строка в таблице PostOffice:

- 731 байт (“чистый”)
- 731 байт (фактический)

Пояснение: “чистый” и фактический объемы совпадают, т.к. в строках таблицы Sending хранятся только domain (business) данные.

Строка в таблице Employee:

- 446 байт (“чистый”)
- 454 байт (фактический)

Строка в таблице StageChangedEvent:

- 124 байт (“чистый”)
- 156 байт (фактический)

“Чистый” объём данных

Пусть S - количество отправлений, E - среднее количество работников, P - количество почтовых отделений, St - среднее количество пройденных этапов на одно отправление.

С определенной долей условности можно считать, что E=200, P=50, St=10.

Тогда “чистый” объём данных в байтах вычисляется по следующей формуле:

$$data_{net} = 731 \cdot P + 446 \cdot E + S \cdot (2012 + St \cdot 124)$$

Или:

$$data_{net} = 125750 + 3252 \cdot S$$

Фактический объём данных

Фактический объём данных в байтах вычисляется по формуле ниже:

$$data_{gross} = 731 \cdot P + 454 \cdot E + S \cdot (2012 + St \cdot 156)$$

Или:

$$data_{gross} = 127350 + 3572 \cdot S$$

Избыточность

Избыточность модели данных можно вычислить по этой формуле:

$$\frac{data_{gross}}{data_{net}} = \frac{127350 + 3572 \cdot S}{125750 + 3252} = 1 + \frac{1600 + 320 \cdot S}{125750 + 3252 \cdot S}$$

Направление роста модели при увеличении количества объектов каждой сущности

При увеличении количества отправлений S, среднего количества работников E, количества почтовых отделений P или среднего количества пройденных этапов на одно отправление St рост модели будет линейным.

Запросы к БД для реализации сценариев использования

“Просмотр состояния отправления”

Запрос (основной сценарий):

```
SELECT name, timestamp, postcode
FROM stage_changed_event WHERE order_id = "123e4567-e89b-12d3-a456-426655440000";
```

Количество запросов: 1

Количество задействованных таблиц: 1

Запрос (альтернативный сценарий):

```
SELECT
FROM sending
WHERE type = "Посылка" AND
      sender_settlement = "пос. Нугуш" AND
      status = "В пути";
```

Количество запросов: 1

Количество задействованных таблиц: 1

“Регистрация нового отправления”

Запрос:

```
INSERT INTO sending(sender_surname, sender_name, sender_middle_name, sender_region, sender_district, sender_settlement, sender_street, sender_building, sender_apartment, sender_postcode,
```

```
                          receiver_surname, receiver_name,
receiver_middle_name, receiver_region, receiver_district,
receiver_settlement, receiver_street, receiver_building,
receiver_apartment, receiver_postcode,
                          type, length, width, height, weight, status)
VALUES
     ("Газизов", "Эмиль", "Наилевич", "Респ. Татарстан", "Альметьевский
район", "г. Альметьевск", "ул. Советская", "147", "", "436123",
      "Мулюкова", "Регина", "Радиковна", "", "", "г. Москва", "ул.
Победы", "9", "", "194505",
      "Бандероль", 20, 10, 5, 1000, "В пути");
```

Количество запросов: 1

Количество задействованных таблиц: 1

“Просмотр статистики по отправлениям”

Запрос (количество отправлений):

```
SELECT sender_settlement, COUNT(*) AS shipments_number
FROM sending
WHERE sender_settlement IN ("г. Мелеуз", "г. Уфа") AND
      type IN ("Посылка", "Письмо")
GROUP BY sender_settlement
```

Количество запросов: 1

Количество задействованных таблиц: 1

Примечание: запрос для “количество получений” аналогичен приведенному выше, за исключением того, что используется receiver_settlement.

Запрос (среднее время в пути с фильтрацией по городам и типам):

```
WITH
    last_stage AS (SELECT order_id, MAX(timestamp) AS
last_stage_time
          FROM stage_changed_event
          GROUP BY order_id),
SELECT sender_settlement, AVG(TIME_TO_SEC(TIMEDIFF(last_stage_time,
registration_date))) AS average_time
FROM sending JOIN last_stage USING(order_id)
WHERE status = "Доставлено"
GROUP BY sender_settlement;
```

Количество запросов: 2

Количество задействованных таблиц: 2

Запрос (средний вес типов отправлений с фильтрацией по городам и типам):

```
SELECT AVG(weight) AS average_weight, type
FROM sending
WHERE sender_settlement IN ("г. Мелеуз", "г. Уфа") AND
      type IN ("Посылка", "Письмо")
```

```
GROUP BY type;
```

Количество запросов: 1

Количество задействованных таблиц: 1

“Просмотр сотрудников”

Запрос:

```
SELECT surname, name, middle_name, gender, birth_date, position,
postcode, phone_number
FROM Employee JOIN PostOffice USING(postcode)
WHERE postcode = "123456" AND
      name = "Райан" AND
      surname = "Гослинг" AND
      middle_name = "Томасович" AND
      gender = "M" AND
      birth_date = ISODate("1980-11-12T00:00:00Z") AND
      position = "Водитель" AND
      phone_number = "88005553535";
```

Количество запросов: 1

Количество задействованных таблиц: 1

3.4. Сравнение моделей

Удельный объем информации

Пусть $E = 200, P = 50, St = 10, S = 100$. Тогда:

- NoSQL:
  - “Чистый”: 451150 байт
  - Фактический: 467350 байт
  - Избыточность: 1.036
- SQL:
  - “Чистый”: 450950 байт
  - Фактический: 484500 байт
  - Избыточность: 1.074

Что лучше?

- Наименьшее количество запросов: NoSQL
- Наименьшее количество задействованных таблиц (коллекции): Ничья
- Наименьшая избыточность: NoSQL

- Вывод: в данном конкретном случае наиболее выгодным решением является использование NoSQL СУБД.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание

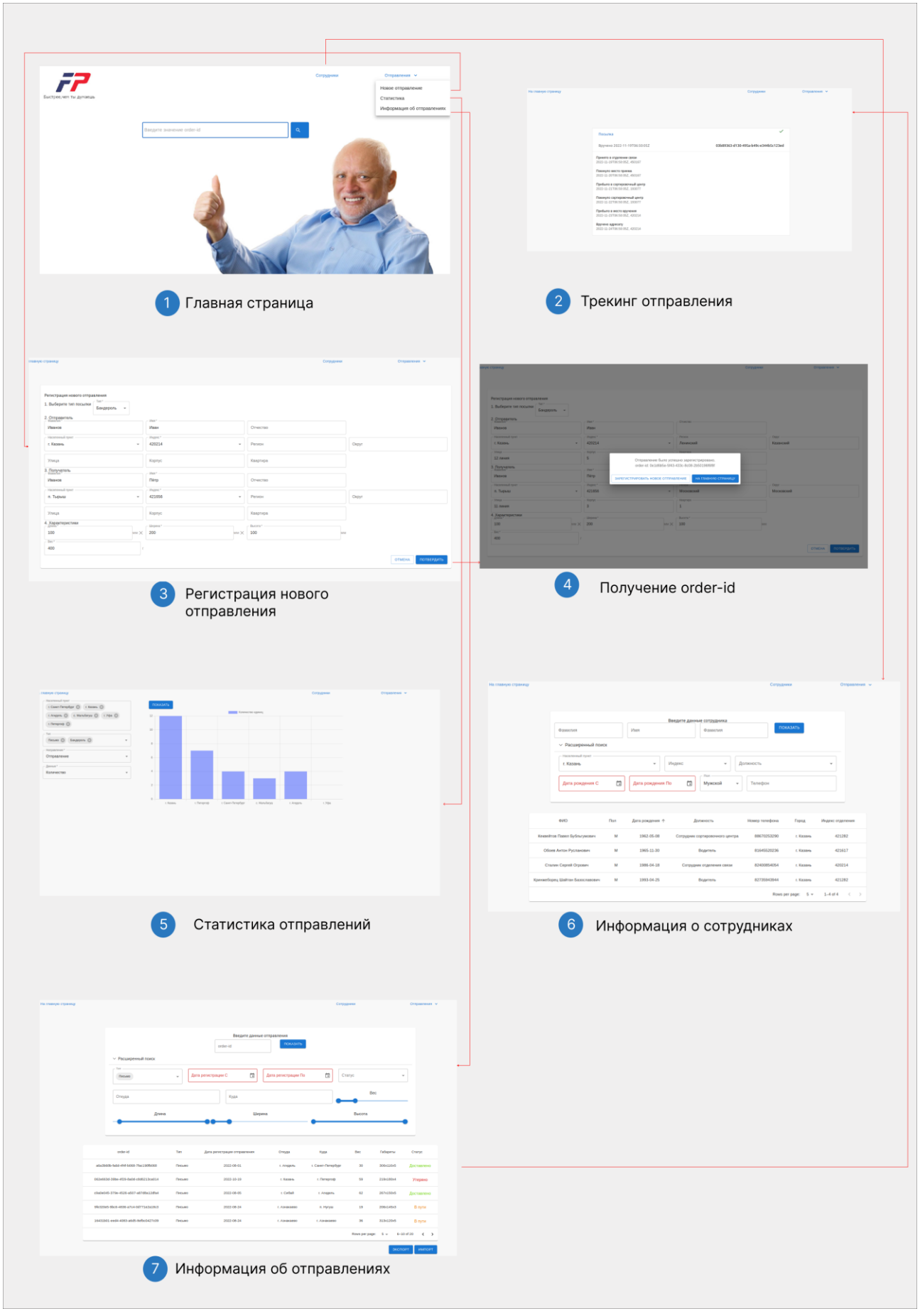
Серверная часть (backend) – Golang-приложение с OpenAPI 3.0 спецификацией. Часть кода сгенерирована по этой спецификации.

Клиентская часть (frontend) – Single Page Application, реализованное с помощью библиотеки React. Клиент использует API, предоставленное серверной частью приложения, для получения и отправки данных.

Для генерации данных на ЯП JavaScript разработана вспомогательная консольная утилита.

Для развертывания приложения используется Docker.

4.2. Схема интерфейса



4.3. Использованные технологии

- БД: MongoDB 6.0.3
- Backend: Golang 1.18
- Frontend: React 18.2, JavaScript
- Deploy: Docker 20.10, Docker Compose 1.25

4.4. Ссылки на Приложение

GitHub репозитории: https://github.com/moevm/nosql2h22-post

ВЫВОД

Достигнутый результат.

В ходе выполнения работы было разработана информационная система для работы почты, в которой моделируется отправка писем, бандеролей и посылок, в виде веб-приложения. В приложении можно создавать отправления, отслеживать их, просматривать списки сотрудников почтовых отделений и статистику по всем отправлениям.

Недостатки и пути для улучшения полученного решения.

В существующем решении имеются следующие недостатки:

1. Отсутствие отдельного микросервиса для моделирования продвижения посылок
2. Импорт/экспорт для данных о сотрудниках и почтовых отделениях.

Первый недостаток не является критичным, т.к. подразумевается, что посылки буду продвигаться в процессе использования системы.

Второй – можно решить дополнением существующего API серверной части и написание соответствующей логики на стороне клиента.

Будущее развитие решения.

Использование нескольких экземпляров БД с репликацией.

Реализация desktop-клиента для ОС Linux и Windows.

ПРИЛОЖЕНИЯ

Документация по сборке и развертыванию приложения

1. Скачать проект из репозитория: https://github.com/moevm/nosql2h22-post
2. Запустить команду docker compose up
3. Открыть приложение по адресу http://localhost:8081

Дополнительную информацию можно найти в README репозитория.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация MongoDB: https://www.mongodb.com/docs/manual/reference/