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: Сервис хранения экспериментов инструмента ripes

Студенты гр. 0304 _____ Голиков А.В.

_____ Решоткин А.С.

_____ Крицын Д.Р.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2023

Студенты
Голиков А.В.
Решоткин А.С.
Крицын Д.Р.
Группа 0304

Тема проекта: Сервис хранения экспериментов инструмента ripes.

Исходные данные:
Необходимо реализовать сервис для хранения экспериментов инструментов ripes с использованием MongoDB.

Содержание пояснительной записки:
«Содержание»
«Введение»
«Качественные требования к решению»
«Сценарий использования»
«Модель данных»
«Разработка приложения»
«Вывод»
«Приложение»

Предполагаемый объем пояснительной записки: Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студенты гр. 0304	_____	Решоткин А.С.
	_____	Голиков А.В.
	_____	Крицын Д.Р.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В качестве индивидуального домашнего задания была выбрана тема “Сервиса хранения экспериментов инструмента ripes”, предполагающая разработку приложения хранения экспериментов с использованием СУБД MongoDB. Исходный код приложения можно найти по ссылке: https://github.com/moevm/nosql2h23-ripes

ANNOTATION

As an individual homework, the topic “Storage services for experiments of the ripes tool” was chosen, which involves the use of the MongoDB. The application source code can be found at: https://github.com/moevm/nosql2h23-ripes

1. ВВЕДЕНИЕ

Цель работы – создать высокопроизводительное и удобное решение для хранения экспериментов инструмента ripes.

Было решено разработать веб-приложение, которое позволит хранить в электронном виде эксперименты инструмента ripes, при этом позволяющее удобно с ними взаимодействовать: импортировать, экспортировать, фильтровать.

2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать приложение с использованием СУБД – MongoDB, в которой будут храниться данные экспериментов ripes, а также с возможностью локального разворота приложения с помощью docker-compose.

3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

3.1. Макеты UI.

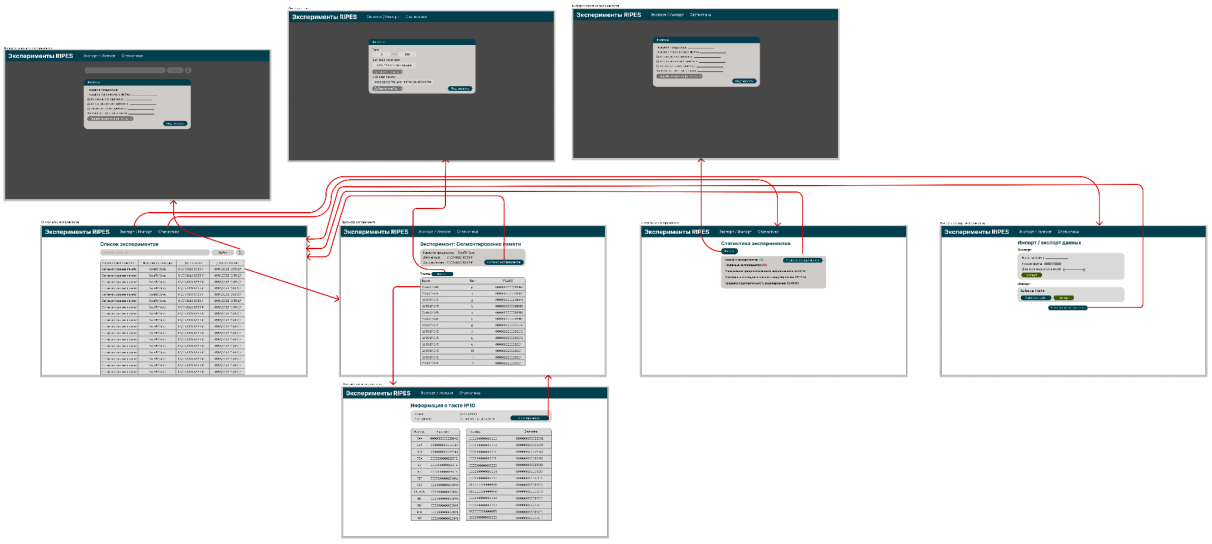

Рис. 1 - Макет UI.

3.2. Описание сценариев использования.

3.2.1. Сценарий использования - “просмотр списка экспериментов”:

Основной сценарий:

1. Система выводит на экран список экспериментов в виде таблицы.
2. Пользователь задаёт фильтрацию для экспериментов и нажимает кнопку “фильтровать”.
3. Система скрывает неподходящие под фильтр эксперименты.

4. Пользователь осуществляет навигацию по списку с использованием прокрутки страницы.

3.2.2. Сценарий использования - “просмотр списка экспериментов”:

Основной сценарий:

1. Пользователь нажимает на эксперимент в списке экспериментов.
2. Система переходит на отдельную веб-страницу, на которой выведена информация об эксперименте: название эксперимента, название процессора, дата начала и окончания выполнения, состояние регистров и памяти.
3. Пользователь выбирает временной штамп, для которого нужно отобразить состояние регистров и памяти.
4. Система обновляет состояние регистров и памяти на странице.

3.2.3. Сценарий использования - “просмотр статистики экспериментов”.

Основной сценарий:

1. Пользователь нажимает на кнопку “статистика экспериментов”.
2. Система переходит на веб-страницу, которая отображает общую статистику экспериментов: количество успешных экспериментов, среднее / минимальное / максимальное время моделирования.

3.2.4. Сценарий использования - “импорт / экспорт данных”.

Основной сценарий:

1. Пользователь нажимает на кнопку “импорт / экспорт”.
2. Система переходит на веб-страницу, которая отображает параметры импорта / экспорта данных: формат файла, диапазон индексов записей, автоматическое наименование экспериментов.
3. Пользователь нажимает на кнопку “импорт” или на кнопку “экспорт”.
4. Система осуществляет импорт или экспорт данных с заданными параметрами.

4. МОДЕЛЬ ДАННЫХ

4.1. Нереляционная модель данных.

4.1.1. Графическое представление модели.

```
{
    _id: <ObjectId1>,
    name: “Сегментирование памяти”,
```

```
     processor: "RV32_5S",
     source_file: "experiment.s",

     start_timestamp: ISODate("2023-10-22T19:00:00Z"),
     end_timestamp: ISODate("2023-10-22T19:00:04Z"),

     instructions_retired: 138,
     cpi: 1.5072463768115942,
     ipc: 0.6634615384615384,
     cycles: 208,

     extensions: "MC",

     Тип данных "конвейер":
     pipeline: {
          _id: <ObjectId3>
          instructions: ["auipc x10 0x10000", "lw x10 0 x10",
"jal x1 28 <fact>", "addi x11 x10 0"],
          cycles: [
               [[0, "IF"]],
               [[0, "ID"], [1, "IF"]],
               [[0, "EX"], [1, "ID"], [2,"IF"]]
          ]
     },

     registers: [
          Тип данных "регистр":
          {
               _id: <ObjectId2>,
               name: "x1",
               value: 28,
               size: 32
          }
```

```
    ]
}
```

4.1.2. Описание назначений коллекций, типов данных и сущностей.

4.1.2.1. Коллекции.

Система имеет одну коллекцию “Эксперименты”, содержащую типы данных “Эксперимент”. При этом тип данных “эксперимент” содержит типы “конвейер” и “регистры”.

4.1.2.2. Типы данных.

- Регистр
    - name - название регистра
    - value - значение регистра по окончанию эксперимента
    - size - размер регистра в битах
- Конвейер
    - instructions - массив уникальных машинных инструкций
    - cycles - история конвейера микроинструкций
- Эксперимент
    - processor - название процессора
    - name - название эксперимента
    - source_file - имя файла с исходным кодом
    - start_timestamp - время начала эксперимента
    - end_timestamp - время окончания эксперимента
    - instructions_retired - количество выполненных инструкций
    - cpi - тактов на одну инструкцию
    - ipc - инструкций на такт
    - cycles - количество тактов в эксперименте
    - extensions - используемые расширения процессора

- pipeline - история конвейера микроинструкций
- registers - окончательные значения регистров процессора

4.1.3. Оценка удельного объема информации, хранимой в модели.

- Тип данных “Регистр”
  - _id - тип ObjectId. V = 12b
  - name - тип String. V = Nrn b, где Nrn ~ 3 - средняя длина имени регистра. V = 3b
  - value - тип Int64. V = 8b
  - size - тип Int. V = 4b Средний размер регистра V = 27b.
- Тип данных “Конвейер”
  - _id - тип ObjectId. V = 12b
  - instructions - тип Array(String). V = Nin * Nasm b, где Nin ~ 150 - среднее количество инструкций в эксперименте, а Nasm ~ 10 - средняя длина машинной инструкции. V = 1500b
  - cycles - Array(Array([Int, Int])). V = (Nm_in + 4) * Ncyc b, где Nm_in ~ 2 - средняя длина микроинструкции, а Ncyc ~ 250 - среднее количество тактов. V = 1500b Средний размер конвейера V = 3012b.
- Тип данных “Эксперимент”
  - _id - тип ObjectId. V = 12b
  - name - тип String. V = Vename b, где Vename ~ 12 - средняя длина названия эксперимента. V = 12b
  - processor - тип String. V = Vcpu b, где Vcpu ~ 6 - средняя длина названия процессора. V = 6b
  - source_file - тип String. V = Vnam b, где Vnam ~ 10 - средняя длина имени файла с исходным кодом. V = 10b
  - start_timestamp - тип Timestamp. V = 8b

- end_timestamp - тип Timestamp. V = 8b
- instructions_retired - тип Int. V = 4b
- cpi - тип Double. V = 8b
- ipc - тип Double. V = 8b
- cycles - тип Int. V = 4b
- extensions - тип String. V = Vext b, где Vext ~ 2 - средняя длина строки дополнений. V = 2b
- pipeline - тип Object. V = 3012b
- registers - тип Array(Object). V = Vreg * 27 b, где Vreg ~ 32 - среднее количество регистров в процессоре. V = 864

Средний размер эксперимента V = 3958b. Объем данных для хранения Nexp экспериментов: V(Nexp) = 3958Nexp. Объем данных для хранения средней по объему базы данных экспериментов: V(200) = 791 600 b ~ 773 Kb.

Избыточность модели Чистый объем данных: 3550Nexp Избыточность модели: 3958Nexp / 3550Nexp = 1,11.

Направление роста модели При увеличении количества экспериментов объём коллекции “Experiments” возрастает линейно.

4.1.4. Примеры запросов к модели

Просмотр списка экспериментов.

- **Получение всех экспериментов**

  db.experiments.findAll()

- **Получение экспериментов по фильтру.**

  db.experiments.findAll({

  name: “Сегментирование памяти”,

```
    processor: ‘RV32_5S’,
      source_file: ‘experiment.s’,
    $expr: {
          { $eq: [
          { $dateDiff: {
              startDate: ‘$start_timestamp’,
              endDate: ‘$end_timestamp’,
              unit: ‘second’
           }},
          10
          ]}
    },
    registers: { name: ‘x1’ }
    })
```

- **Получение информации об эксперименте.**

```
db.experiments.findOne({
      _id: <ObjectId1>
})
```

- **Получение статистики экспериментов.**

```
db.experiments.aggregate([
    { $group: {
          $avg: {
           $dateDiff: {
              startDate: ‘$start_timestamp’,
              endDate: ‘$end_timestamp’,
              unit: ‘second’
          }
```

```
},
$min: {
        $dateDiff: {
            startDate: '$start_timestamp',
            endDate: '$end_timestamp',
            unit: 'second'
        }
},
$max: {
        $dateDiff: {
            startDate: '$start_timestamp',
            endDate: '$end_timestamp',
            unit: 'second'
        }
}
}}
])
```

4.2. Реляционная модель.

4.2.1. Графическое представление модели.

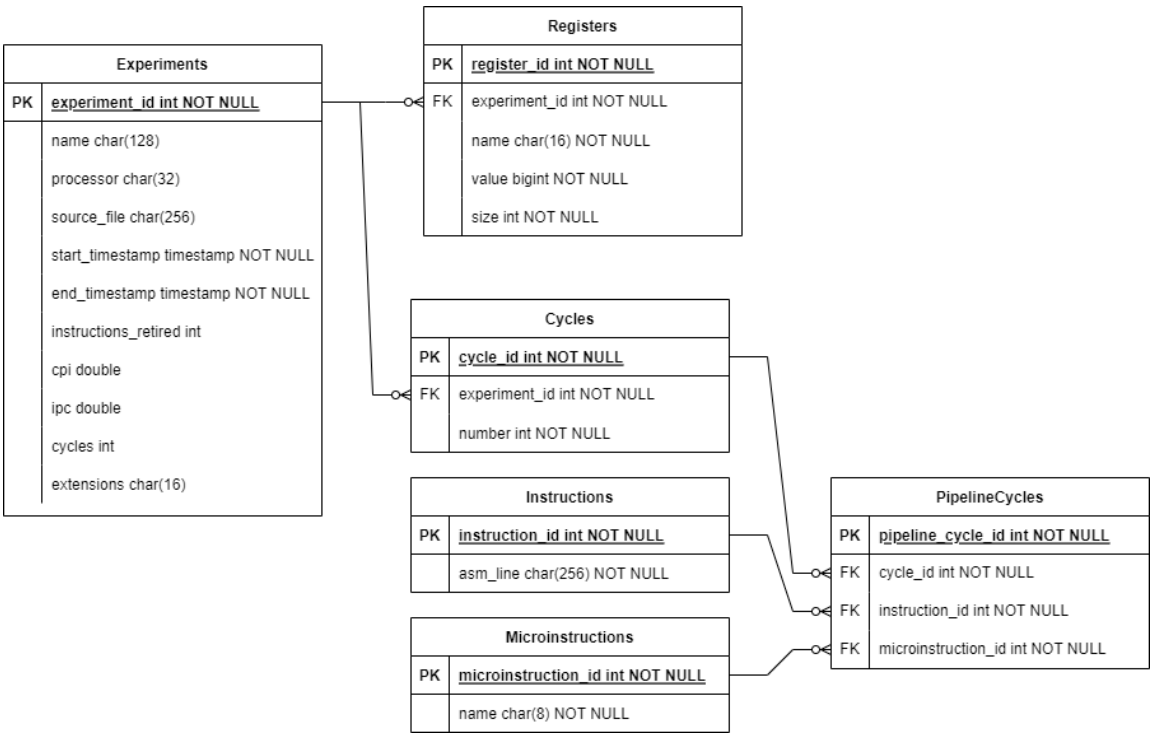

Рис.2. ER - диаграмма реляционной модели.

4.2.2. Описание назначений типов данных и сущностей.

Сущность “Experiment”.

- experiment_id - уникальный идентификатор эксперимента. Тип int. V = 4b.
- name - название процессора. Тип char(32). V = 32b.
- processor - название эксперимента. Тип char(128). V = 128b.
- source_file - имя файла с исходным кодом. Тип char(256). V=256b.
- start_timestamp - время начала эксперимента. Тип timestamp. V=8b.
- end_timestamp - время окончания эксперимента. Тип timestamp. V=8b.
- instruction_retired - время окончания эксперимента. Тип int. V=4b.
- cpi - количество тактов на одну инструкцию. Тип double. V=8b.
- ipc - количество инструкций на такт. Тип double. V = 8b.
- cycles- количество тактов в эксперименте. Тип int. V=4b.

- extensions - используемые расширения процессора. Тип char(16). V=16b.

Размер сущности “Experiment” = 732b

Сущность “Register”.

- register_id - уникальный идентификатор регистра. Тип int NOT NULL. V = 4b.
- experiment_id - уникальный идентификатор эксперимента. Тип int NOT NULL. V = 4b.
- name - название регистра. Тип char(16). V = 16b.
- value - значение регистра по окончанию эксперимента. Тип bigint. V = 8b.
- size -размер регистра в битах. Тип int. V = 4b.

Размер сущности “Register” = 36b
Сущность “Cycle”.

- cycle_id - уникальный идентификатор такта. Тип int. V = 4b.
- experiment_id - уникальный идентификатор эксперимента. Тип int. V = 4b.
- number - номер такта. Тип int. V = 4b.

Размер сущности “Cycle” = 12b.

Сущность “Instruction”.

- instruction_id - уникальный идентификатор инструкции. Тип char. V = 8b.
- asm_line - машинная инструкция. Тип char(256). V = 256b.
- Размер сущности “Instruction” = 264b.
- Сущность “Microinstruction”.

- microinstruction_id - уникальный идентификатор микроинструкции. Тип int. V = 4b.
- name - название микроинструкции. Тип char(8). V = 8b.

Размер сущности “Microinstruction” = 12b.

Сущность “PipelineCycle”.

- pypeline_cycle_id - уникальный идентификатор. Тип int. V = 4b.
- cycle_id - уникальный идентификатор такта. Тип int. V = 4b.
- instruction_id - уникальный идентификатор инструкции. Тип int. V = 4b.
- microinstruction_id - уникальный идентификатор микроинструкции. Тип int. V = 4b.

Размер сущности “PipelineCycle” = 16b.

Среднее количество сущностей “Register” на один эксперимент = 32.

Среднее количество сущностей “Cycle” на один эксперимент = 250.

Среднее количество сущностей “Instruction” на один эксперимент = 150.

Среднее количество сущностей “Microinstruction” на один эксперимент = 30.

Среднее количество сущностей “PipelineCycle” на один такт = 5.

Объем данных для хранения Nexp экспериментов:

V(Nexp) = (32*36* + *250*12 + 150*264* + *30*12 + 250516 + 732)Nexp = 64844Nexp

Объем данных для хранения средней по объему базы данных экспериментов:

V(200) = 12968800b ~ 12Mb.

4.2.3. Примеры запросов к модели.

4.2.3.1. Просмотр списка экспериментов

```
SELECT `Experiments`.experiment_id, `Experiments`.name,
`Experiments`.processor, `Experiments`.start_timestamp,
`Experiments`.end_timestamp FROM `Experiments` JOIN
`Registers` USING (experiment_id) WHERE `Experiments`.name =
'Сегментирование памяти' AND `Experiments`.source_file =
'experiment.s' AND
TIMESTAMPDIFF(`Experiments`.start_timestamp,
`Experiments`.end_timestamp, SECOND) = 3 AND `Registers`.name
= 'x1';
```

Количество запросов к БД равно V(Nexp) = 32Nexp, где Nexp - количество документов в коллекции “Experiments” (32 - среднее количество регистров в эксперименте).

4.2.3.2. Просмотр информации об эксперименте

```
SELECT * FROM `Experiments` WHERE
`Experiments`.experiment_id=3 JOIN `Register` USING
(experiment_id) JOIN `Cycle` USING (experiment_id) JOIN
`PipelineCycle` USING (cycle_id) JOIN `Instructions` USING
(instruction_id) JOIN `Microinstructions` USING
(microinstruction_id);
```

Количество запросов к БД равно V(Nexp) = 322505*(150+30)Nexp = 7200000Nexp, где Nexp - количество документов в коллекции “Experiments”.

4.2.3.3. Просмотр статистики экспериментов

SELECT AVG(TIMESTAMPDIFF(`Experiments`.start_timestamp, `Experiments`.end_timestamp, SECOND)) AS duration_avg,

MIN(TIMESTAMPDIFF(`Experiments`.start_timestamp, `Experiments`.end_timestamp, SECOND)) AS duration_min, MAX(TIMESTAMPDIFF(`Experiments`.start_timestamp, `Experiments`.end_timestamp, SECOND)) AS duration_max FROM ‘Experiments’

Количество запросов к БД равно V(Nexp) = Nexp.

4.3. Сравнение моделей.

- SQL модель данных требует больше места из-за наличия связующих таблиц, а также большего количества сущностей (что затрачивает дополнительную память на идентификаторы) и зарезервированной под строки памяти. Одна и та же база данных на 200 экспериментов имеет размер 773 килобайта в MongoDB, а в SQL - 12 мегабайт.
- SQL модель требует большего количества запросов из-за необходимости множественных соединений различных таблиц:
- Для запроса “Просмотр списка экспериментов” в SQL требуется в среднем в 32 раза больше обращений к БД, чем в MongoDB из-за необходимости соединения таблиц “Experiment” и “Register”.
- Для запроса “Просмотр информации об эксперименте” в SQL требуется в среднем 7200000 раз больше обращений к БД, чем в MongoDB, что обусловлено необходимостью соединения всех таблиц в БД.
- Для запроса “Просмотр статистики экспериментов” требуется одинаковое количество запросов из-за отсутствия необходимости соединения таблиц

4.4. Вывод сравнения моделей.

Таким образом, MongoDB гораздо лучше подходит для данной задачи, поскольку как объем данных, так и количество запросов гораздо меньше в MongoDB, чем в SQL.

5. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

5.1 Краткое описание.

Приложение имеет клиент-серверную архитектуру и использует подход REST.

Приложение разработано на языке программирования JavaScript. Клиентская часть реализована при помощи фреймворка Vue 3, серверная часть - при помощи фреймворка express.

Приложение разворачивается при помощи платформы докер, используя три отдельных виртуальных контейнера для клиентской части приложения, серверной части приложения и базы данных.

5.2. Снимки экрана приложения.

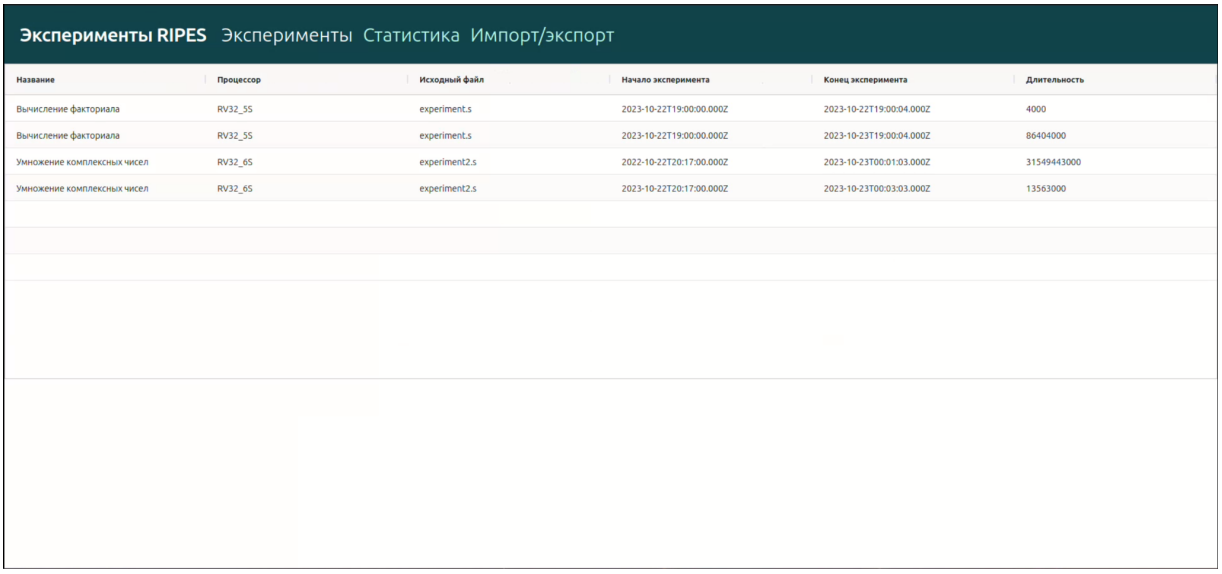

Рис. 3. Страница “Список экспериментов”.

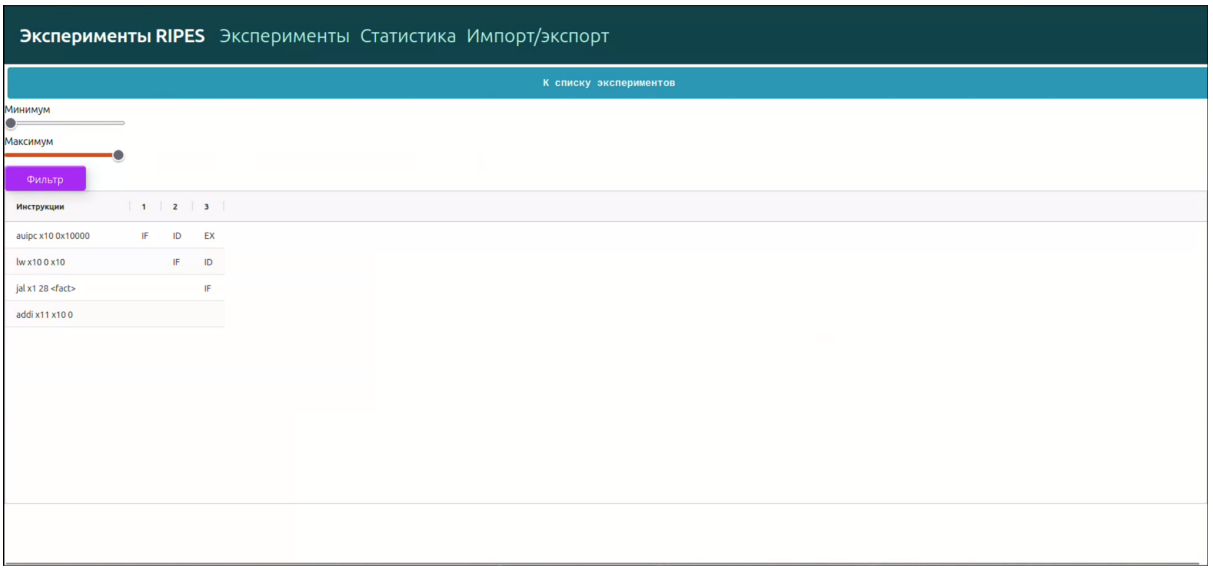

Рис. 4. Страница “Эксперимент”.

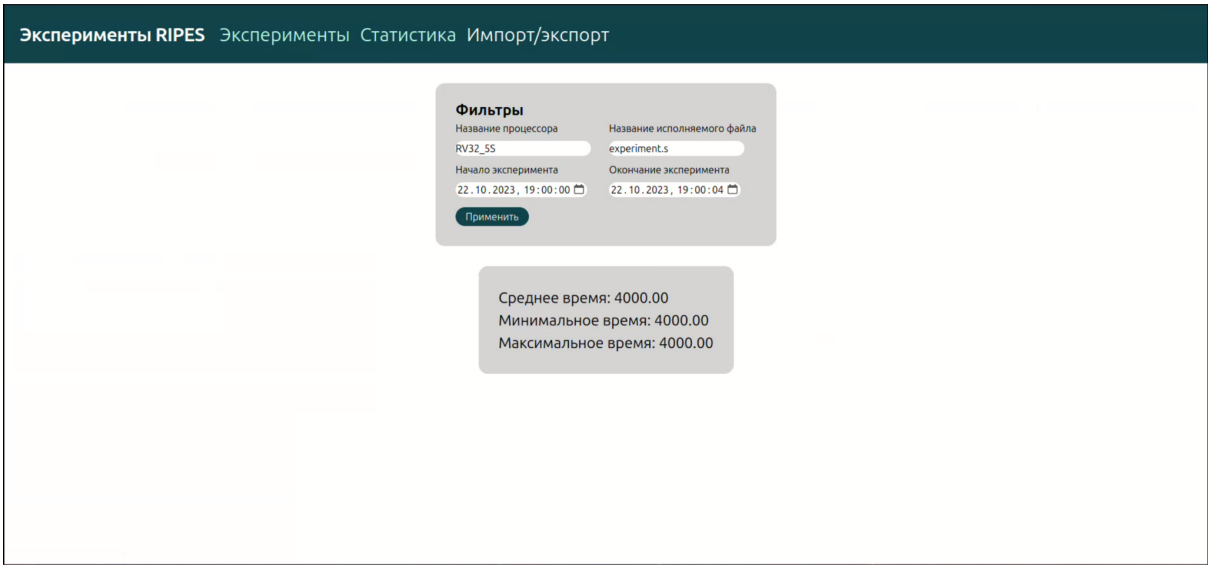

Рис. 5. Страница “Статистика экспериментов”.

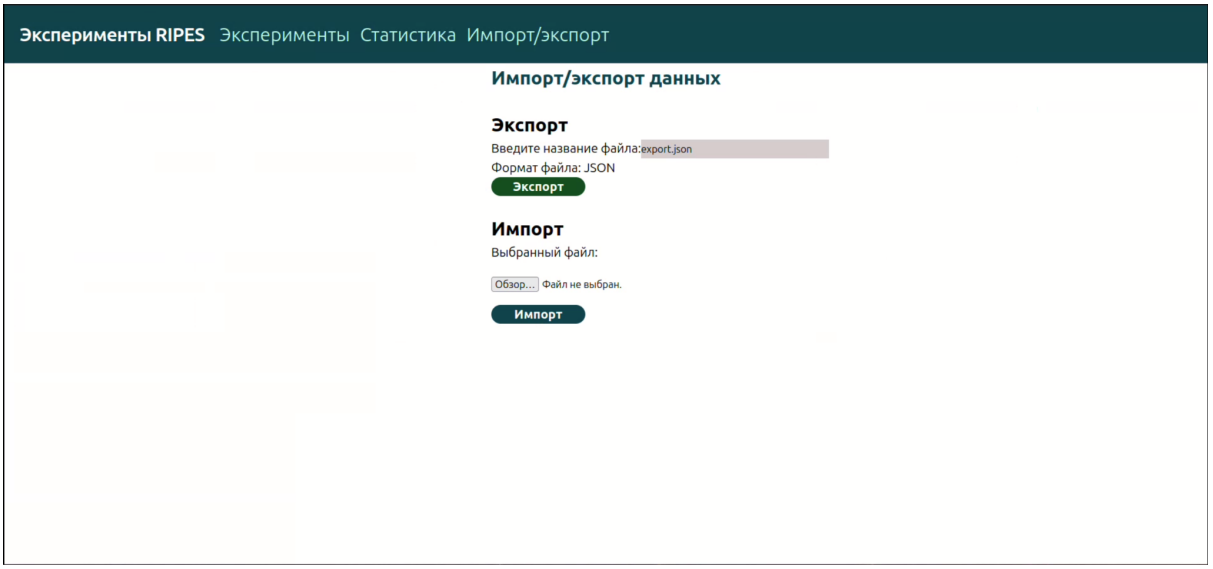

Рис. 6. Страница “Импорт / экспорт данных”.

6. ВЫВОДЫ.

6.1 Достигнутые результаты

В ходе работы было разработано приложение веб-сервиса хранения экспериментов инструмента ripes с использованием нереляционной базы данных MongoDB. Была реализована вся требуемая функциональность, такая как фильтрация, импорт, экспорт файлов, просмотр статистика экспериментов. Также была добавлена возможность локального разворота приложения с помощью docker-compose.

6.2 Недостатки и пути для улучшения полученного решения.

Недостатком приложения является примитивный дизайн и функционал. Пока что, из возможных функций доступны только импорт/экспорт файлов, просмотр экспериментов и их фильтрация. В перспективе развития приложения возможно реализовать гораздо более широкий функционал: интеграция с другими сервисами или платформами, реализация инструментов аналитики и визуализации результатов экспериментов, реализация возможности сохранения данных в облаке.

6.3 Будущее развитие решения

Улучшение функциональности сервиса путем реализации различных возможностей:

Интеграция с облачными сервисами для сохранения данных в облаке.

Интеграция искусственного интеллекта в качестве инструмента аналитики и визуализации результатов экспериментов.