

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

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ
по дисциплине «Введение в нереляционные системы управления базами
данных»
Тема: Хаб для ML датасетов

Студент гр. 2383	_____	Борисов И. В.
Студент гр. 2304	_____	Карпунин К. А.
Студент гр. 2381	_____	Соколов С. А.
Студент гр. 2383	_____	Сыздыков Н. К.
Студент гр. 2381	_____	Тищенко А. М.
Преподаватель	_____	Заславский М. М.

Санкт-Петербург

2025

ЗАДАНИЕ

Студенты

Борисов И. В., гр. 2383

Карпунин К. А., гр. 2304

Соколов С. А., гр. 2381

Сыздыков Н. К., гр. 2383

Тищенко А. М., гр. 2381

Тема работы: Хаб для ML датасетов

Исходные данные:

разработать информационную систему для хранения и визуализации набора датасетов для ML.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Выводы»

«Приложения»

«Литература»

Предполагаемый объем пояснительной записки:

Не менее 50 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студент гр. 2383

Студент гр. 2304

Студент гр. 2381

Студент гр. 2383

Студент гр. 2381

Преподаватель

Борисов И. В.

Карпунин К. А.

Соколов С. А.

Сыздыков Н. К.

Тищенко А. М.

Заславский М. М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема разработки информационной системы для хранения и визуализации набора датасетов для ML с использованием MongoDB в качестве СУБД. Найти исходный код и всю дополнительную информацию можно по ссылке:

<https://github.com/moevm/nosql1h25-mldata>

ANNOTATION

As part of this course, it was supposed to develop an application in a team on one of the set topics. The topic of developing an information system for storing and visualizing a set of ML datasets using MongoDB as a DBMS was chosen. You can find the source code and all additional information here:

<https://github.com/moevm/nosql1h25-mldata>

СОДЕРЖАНИЕ

1.	Введение	7
2.	Сценарии использования	9
3.	Модель данных	12
4.	Разработанное приложение	45
5.	Выводы	52
6.	Приложения	54
7.	Список использованных источников	57

1. ВВЕДЕНИЕ

1.a. Актуальность решаемой проблемы

Веб-приложение для хранения ML-датасетов с использованием MongoDB актуально благодаря растущим потребностям в эффективном управлении большими объемами данных, характерных для машинного обучения. MongoDB, как документоориентированная СУБД, обеспечивает гибкость хранения неструктурированных данных, таких как изображения, тексты или временные ряды, а также метаданных, включая аннотации и версии, что упрощает масштабирование и адаптацию под изменяющиеся требования проектов. Централизованная веб-платформа устраняет проблему фрагментации данных, обеспечивая совместный доступ, контроль версий и воспроизводимость экспериментов, что критично для исследователей и инженеров.

1.b. Постановка задачи

Разработка информационной системы для хранения и визуализации набора ML датасетов.

1.c. Предлагаемое решение

Приложение будет иметь клиент-серверную архитектуру: веб-интерфейс взаимодействует с backend-частью, написанной на Python с применением фреймворка Flask.

Клиентская сторона будет включать несколько разделов:

- стартовую страницу с карточками наборов данных, где отображаются название, описание, объём и размер коллекции;
- отдельный интерфейс для детального просмотра выбранного информационного массива с визуализацией и опцией скачивания в CSV/JSON;

- форму для добавления новых записей с ручным вводом метаданных или загрузкой файлов;
- а также инструмент пакетной обработки для одновременного импорта/экспорта групп данных.

Серверная часть обрабатывает запросы, извлекает и сохраняет информацию в MongoDB. Гибкость структуры базы данных позволяет хранить как метаинформацию (описания, статистику), так и сами файлы, обеспечивая совместимость с разнородными форматами данных.

1.d. Качественные требования к решению

Требуется разработать приложение, которое корректно взаимодействует с MongoDB и отвечает на запросы пользователя.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.a. Макет UI

Макет доступен по ссылке в [репозитории проекта](#)

2.b. Сценарии использования

Действующее лицо -- пользователь.

- Страница входа
 - Зарегистрироваться в сервисе
 - Имя пользователя
 - Ввести логин
 - Ввести пароль
 - Подтвердить пароль
 - Войти в сервис
 - Ввести логин
 - Ввести пароль

На всех следующих страницах отображается логотип сервиса, который при нажатии переводит на главную страницу.

- Страница "Главная" (любой авторизованный пользователь, если не указано обратного)
 - отфильтровать датасеты по названию (регистронезависимый поиск), размеру (верхняя и нижняя граница), количеству обучающих данных (верхняя и нижняя граница), количеству загрузок (верхняя и нижняя граница), количеству просмотров (верхняя и нижняя граница), дате добавления (дата не ранее и дата не позднее)
 - отсортировать датасеты по названию, размеру, дате добавления, количеству обучающих данных, количеству загрузок, количеству просмотров (по убыванию/возрастанию)
 - перейти на страницу "Просмотр датасета"

- перейти на страницу "Добавить датасет"
- перейти на страницу пользователя
- Страница "Добавить датасет" (любой авторизованный пользователь)
 - Ввести название датасета
 - Ввести описание датасета
 - Загрузить файл / ввести данные датасета (зависит от выбора в dropdown меню)
 - Выбрать тип файла
 - Сохранить датасет
 - Отменить загрузку
- Страница "Просмотр датасета" (любой авторизованный пользователь)
 - Просмотр информации датасета
 - Описание
 - Характеристики
 - Просмотр графиков распределений и данных датасета / дэшбордов (зависит от выбора в dropdown меню)
 - Управление датасетом
 - Скачать датасет
 - Перейти на страницу редактирования датасета (обычный пользователь имеет доступ к редактированию только своих датасетов)
 - Удалить датасет (обычный пользователь имеет доступ к удалению только своих датасетов)
 - Активность
 - Статистики по просмотрам датасета
 - Статистики по скачиванием датасета
- Страница "Редактировать датасет" (любой авторизованный пользователь)
 - Изменить название датасета
 - Изменить описание датасета

- Изменить данные датасета -- загрузить файл / ввести данные датасета (зависит от выбора в dropdown меню)
- Изменить формат файла
- Вернуться на страницу "Главная"
- Сохранить датасет
- Отменить редактирование
- Страница пользователя
 - Просмотреть информацию о профиле пользователя: имя пользователя, логин, количество созданных датасетов, дата создания редактирования, дата последнего редактирования аккаунта
 - Изменить имя пользователя
 - Изменить пароль
 - Статистика по загруженным датасетам
 - Выйти из аккаунта

Действующее лицо -- админ (одна учетная запись). СИ остается таким же, дополнительных окон для данной роли не предусмотрено.

- На странице пользователя добавляются кнопки по массовому импорту/экспорту
- На странице пользователя добавляются кнопки по поиску всех пользователей, зарегистрированных в системе и их блокировке/разблокировке.
- Админ может удалять и редактировать все датасеты.

3. МОДЕЛЬ ДАННЫХ

3.а. Нереляционная модель данных

3.а.І. Графическое представление модели в виде JSON

```
{
  "DatasetInfoCollection": [
    {
      "_id": "UUID", // Обязательное уникальное поле; UUID будет указываться при добавлении объекта в коллекцию
                    // Объекты DatasetInfo и DatasetActivity, соответствующие одному датасету, будут иметь одинаковый UUID

      "name": "string", // Название датасета
      "description": "string", // Описание датасета
      "creationDate": "ISODate", // Дата загрузки датасета
      "author": "string", // Никнейм пользователя-создателя
      "rowCount": "int32", // Количество строк в датасете
      "columnCount": "int32", // Количество колонок в датасете
      "size": "int32", // Размер файла датасета в байтах
      "path": "string", // Абсолютный путь к файлу датасета
      "lastVersionNumber": "int32", // Инкрементируется при изменениях
      "lastModifiedDate": "ISODate", // Дата последнего обновления
      "lastModifiedBy": "string" // Никнейм редактора
    }
  ],

  "DatasetActivityCollection": [
    {
      "_id": "UUID", // Обязательное уникальное поле

      "statistics": {
        "YYYY-MM-DD_1": { // Ключом выступает дата в формате YYYY-MM-DD
          "views": "int32", // Число просмотров в указанную дату
          "downloads": "int32" // Число скачиваний в указанную дату
        },

        "...", // "statistic" содержит такие объекты для последних 30 дней

        "YYYY-MM-DD_30": {
          "views": "int32",
          "downloads": "int32"
        }
      }
    }
  ],

  "DatasetGraphsCollection": [
    {
      "_id": "UUID", // Обязательное уникальное поле

      "graphs": [
        {
          "name": "string", // Название графика
          "data": "BinData" // Изображение в бинарном формате
        },

        "..." // "graphs" содержит все графики датасета
      ]
    }
  ],

  "UserCollection": [
    {
      "_id": "UUID", // Обязательное уникальное поле
      "username": "string", // Никнейм пользователя
      "login": "string", // Логин пользователя
      "password": "string", // Пароль пользователя
      "status": "int32", // Статус пользователя; 0 - администратор, 1 - активен, 2 - заблокирован
      "createdDatasetsCount": "int32", // Количество созданных датасетов
      "accountCreationDate": "ISODate", // Дата регистрации
      "lastAccountModificationDate": "ISODate" // Дата последнего изменения профиля
    }
  ]
}
```

Рисунок 1: Графическое представление модели в виде JSON

В данном проекте используется документноориентированная СУБД MongoDB. В базе данных созданы четыре коллекции:

- DatasetInfoCollection - Коллекция, содержащая метаданные датасета;
- DatasetActivityCollection - Коллекция, содержащая статистику просмотров и скачиваний за последние 30 дней;
- DatasetGraphsCollection - Коллекция, содержащая графики, визуализирующие данные датасета;
- UserCollection - Коллекция, содержащая данные о пользователях.

Каждая из коллекций описана более подробно ниже (вместо значений полей указан их тип).

3.а.И. Описание назначений коллекций, типов данных и сущностей

Коллекция DatasetInfoCollection

Данная коллекция хранит объекты DatasetInfo - объекты, содержащие информацию о датасете, такую как название, описание, размер и проч.

```
"DatasetInfo": {
  "_id": "UUID",                // UUIDv4
  "name": "string",             // Строка, ограниченная 50
  СИМВОЛАМИ
  "description": "string",      // Строка, ограниченная 1000
  СИМВОЛОВ
  "creationDate": "ISODate",    // Дата в формате "YYYY-MM-DDTHH:MM:SSZ"
  "author": "string",          // Строка, ограниченная 50
  СИМВОЛАМИ
  "rowCount": "int32",         // Целочисленное число
  "columnCount": "int32",      // Целочисленное число
  "size": "int32",             // Целочисленное число
  "path": "string",            // Строка, ограниченная 1024
  СИМВОЛАМИ
  "lastVersionNumber": "int32", // Целочисленное число
  "lastModifiedDate": "ISODate", // Дата в формате "YYYY-MM-DDTHH:MM:SSZ"
  "lastModifiedBy": "string"    // Строка, ограниченная 50
  СИМВОЛАМИ
}
```

Коллекция DatasetActivityCollection

Данная коллекция хранит объекты DatasetActivity - объекты, содержащие статистику просмотров и скачиваний за последние 30 дней.

```
"DatasetActivity": {
  "_id": "UUID", // UUIDv4
  "statistics": {
    "YYYY-MM-DD_1": { // Дата в формате "YYYY-MM-DD"
      "views": "int32", // Число просмотров в указанную
дату
      "downloads": "int32" // Число скачиваний в указанную
дату
    },
    "...",
    "YYYY-MM-DD_30": {
      "views": "int32",
      "downloads": "int32"
    }
  }
}
```

Коллекция DatasetGraphsCollection

Данная коллекция хранит объекты DatasetGraphs - объекты, содержащие графики, визуализирующие данные датасета.

```
"DatasetGraphs": {
  "_id": "UUID", // UUIDv4
  "graphs": [
    {
      "name": "string", // Строка, ограниченная 50
символами
      "data": "BinData" // Бинарные данные
    },
    "...",
  ]
}
```

Коллекция UserCollection

Данная коллекция хранит объекты User - объекты, содержащие информацию о пользователях, такую как имя, логин, пароль, статус и проч.

```
"User": {
  "_id": "UUID",                // UUIDv4
  "username": "string",         // Строка, ограниченная 50
символами
  "login": "string",            // Строка, ограниченная 50
символами
  "password": "string",         // Строка, ограниченная 50
символами
  "status": "int32",            // Целочисленное число
  "createdDatasetsCount": "int32", // Целочисленное число
  "accountCreationDate": "ISODate", // Дата в формате "YYYY-MM-DDTHH:MM:SSZ"
  "lastAccountModificationDate": "ISODate" // Дата в формате "YYYY-MM-DDTHH:MM:SSZ"
}
```

3.а.III. Оценка удельного объема информации, хранимой в модели

При добавлении нового датасета создаются объекты DatasetInfo, DatasetActivity и DatasetGraphs. При добавлении нового пользователя создается объект User.

Оценка размера объекта DatasetInfo:

```
"DatasetInfo": {
  "_id": "UUID",                // 36 байт
  "name": "string",             // 50 байт
  "description": "string",      // 1000 байт
  "creationDate": "ISODate",    // 8 байт
  "author": "string",           // 50 байт
  "rowCount": "int32",          // 4 байта
  "columnCount": "int32",       // 4 байта
  "size": "int32",              // 4 байта
}
```

```

    "path": "string", // 1024 байта
    "lastVersionNumber": "int32", // 4 байта
    "lastModifiedDate": "ISODate", // 8 байт
    "lastModifiedBy": "string" // 50 байт
}

```

Итого: 2242 байт

Оценка размера объекта DatasetActivity:

```

"DatasetActivity": {
    "_id": "UUID", // 36 байт
    "statistics": { // Общий размер "statistic": 30
        * (4 + 4) = 240 байт
        "YYYY-MM-DD_1": {
            "views": "int32", // 4 байта
            "downloads": "int32" // 4 байта
        },

        "...",

        "YYYY-MM-DD_30": {
            "views": "int32",
            "downloads": "int32"
        }
    }
}

```

Итого: 276 байт

Оценка размера объекта DatasetGraphs:

```

"DatasetGraphs": {
    "_id": "UUID", // 36 байт
    "graphs": [ // Общий размер "graphs": 7 *
        (50 + 102'400) = 717'150 байт
        // Предполагается, что на
        датасет в среднем приходится 7 графиков
        {
            "name": "string", // 50 байт

```



```

        "data": "BinData"                                // 100 * 1024 байт
                                                         // Предполагается, что средний
размер изображения составляет 100 кбайт
    },
    "...",
]
}

```

Итого: 717'186 байт

Таким образом, при добавлении одного датасета в БД необходимо выделить $2242 + 276 + 717'186 = 719'704$ байт ≈ 703 кбайт.

Оценка размера объекта User:

```

"User": {
    "_id": "UUID",                                // 36 байт
    "username": "string",                        // 50 байт
    "login": "string",                          // 50 байт
    "password": "string",                      // 50 байт
    "status": "int32",                          // 4 байта
    "createdDatasetsCount": "int32",            // 4 байта
    "accountCreationDate": "ISODate",           // 8 байт
    "lastAccountModificationDate": "ISODate"    // 8 байт
}

```

Итого: 210 байт

Таким образом, при добавлении одного пользователя в БД необходимо выделить 210 байт.

Общая оценка:

А значит весь объем информации в БД занимает $(703 * N + 0.2 * M)$ кбайт, где N - количество датасетов в системе, а M - количество пользователей в системе.

Предположим, что на один датасет приходится 10 пользователей (по сравнению с количеством датасетов количество пользователей в разы больше, так как большинство пользователей не загружают новые датасеты, а пользуются уже существующими).

Соответственно, $M = 10 * N$

Тогда получаем объем информации в зависимости от количества датасетов:

$$V(N) = 703 * N + 0.2 * (10 * N) = (703 + 2) * N = 705 * N \text{ кбайт}$$

Избыточность данных

Оценка чистого объема данных объекта DatasetInfo:

```
"DatasetInfo": {  
    "name": "string",                // 50 байт  
    "description": "string",         // 1000 байт  
    "creationDate": "ISODate",       // 8 байт  
    "rowCount": "int32",             // 4 байта  
    "columnCount": "int32",          // 4 байта  
    "size": "int32",                 // 4 байта  
    "path": "string",                // 1024 байта  
    "lastVersionNumber": "int32",    // 4 байта  
    "lastModifiedDate": "ISODate",   // 8 байт  
}
```

Итого: 2106 байт

Оценка чистого объема данных объекта DatasetActivity:

```
"DatasetActivity": {  
    "statistics": {                  // Общий размер "statistic": 30  
        * (4 + 4) = 240 байт  
        "YYYY-MM-DD_1": {  
            "views": "int32",        // 4 байта  
            "downloads": "int32"     // 4 байта  
        },  
        "..."
```

```

        "YYYY-MM-DD_30": {
            "views": "int32",
            "downloads": "int32"
        }
    }
}

```

Итого: 240 байт

Оценка чистого объема данных объекта DatasetGraphs:

```

"DatasetGraphs": {
    "graphs": [
(50 + 102'400) = 717'150 байт
// Общий размер "graphs": 7 *
// Предполагается, что на
датасет в среднем приходится 7 графиков
{
    "name": "string",
    "data": "BinData"
// 50 байт
// 100 * 1024 байт
// Предполагается, что средний
размер изображения составляет 100 кбайт
},
    "...
]
}

```

Итого: 717'150 байт

Таким образом, при добавлении одного датасета в БД необходимо выделить $2106 + 240 + 717'150 = 719'496$ байт ≈ 702 кбайт.

Оценка чистого объема данных объекта User:

```

"User": {
    "username": "string",
    "login": "string",
    "password": "string",
// 50 байт
// 50 байт
// 50 байт

```

```
"status": "int32", // 4 байта
"createdDatasetsCount": "int32", // 4 байта
"accountCreationDate": "ISODate", // 8 байт
"lastAccountModificationDate": "ISODate" // 8 байт
}
```

Итого: 174 байт

Таким образом, при добавлении одного пользователя в БД необходимо выделить 174 байта.

Общая оценка:

А значит весь чистый объем данных в БД составляет $(702 * N + 0.1 * M)$ кбайт, где N - количество датасетов в системе, а M - количество пользователей в системе.

Предположим, что на один датасет приходится 10 пользователей (по сравнению с количеством датасетов количество пользователей в разы больше, так как большинство пользователей не загружают новые датасеты, а пользуются уже существующими).

Соответственно, $M = 10 * N$

Тогда получаем чистый объем данных в зависимости от количества датасетов:

$$V_c(N) = 702 * N + 0.1 * (10 * N) = (702 + 1) * N = 703 * N \text{ кбайт}$$

В таком случае, коэффициент избыточности будет равен:

$$C = V(N) / V_c(N) = (705 * N) / (702 * N) \approx 1.004$$

Направление роста модели при увеличении количества объектов каждой сущности.

При увеличении количества объектов любой из сущностей модель будет расти линейно.

$$V(N, M) = 703 * N + 0.2 * M \text{ кбайт,}$$

где N - количество датасетов в системе, а M - количество пользователей в системе.

Примеры данных

Объект DatasetInfo:

```
"DatasetInfo": {
  "_id": "a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8",
  "name": "Sales Data 2023",
  "description": "This dataset contains information about sales",
  "creationDate": ISODate("2023-07-15T09:30:45Z"),
  "author": "John Sales",
  "rowCount": 12500,
  "columnCount": 24,
  "size": 5242880,
  "path": "/datasets/sales/dataset.csv",
  "lastVersionNumber": 5,
  "lastModifiedDate": ISODate("2023-09-20T14:15:22Z"),
  "lastModifiedBy": "Jane Sales"
}
```

Объект DatasetActivity:

```
"DatasetActivity": {
  "_id": "b2c3d4e5-f6g7-8910-h1i2-j3k4l5m6n7o8",
  "statistics": {
    "2023-10-01": {
      "views": 42,
      "downloads": 7
    },
    "2023-10-02": {
      "views": 38,
      "downloads": 5
    },
    "2023-10-03": {
      "views": 55,
      "downloads": 12
    },
    "2023-10-04": {
      "views": 61,
```

```
    "downloads": 9
  },
  "2023-10-05": {
    "views": 47,
    "downloads": 8
  },
  "2023-10-06": {
    "views": 72,
    "downloads": 15
  },
  "2023-10-07": {
    "views": 89,
    "downloads": 21
  },
  "2023-10-08": {
    "views": 65,
    "downloads": 14
  },
  "2023-10-09": {
    "views": 53,
    "downloads": 11
  },
  "2023-10-10": {
    "views": 48,
    "downloads": 7
  },
  "2023-10-11": {
    "views": 57,
    "downloads": 10
  },
  "2023-10-12": {
    "views": 62,
    "downloads": 13
  },
  "2023-10-13": {
    "views": 71,
    "downloads": 16
  },
  "2023-10-14": {
    "views": 84,
    "downloads": 19
  }
```

```
},
"2023-10-15": {
  "views": 76,
  "downloads": 18
},
"2023-10-16": {
  "views": 59,
  "downloads": 12
},
"2023-10-17": {
  "views": 63,
  "downloads": 11
},
"2023-10-18": {
  "views": 67,
  "downloads": 14
},
"2023-10-19": {
  "views": 72,
  "downloads": 16
},
"2023-10-20": {
  "views": 81,
  "downloads": 20
},
"2023-10-21": {
  "views": 93,
  "downloads": 25
},
"2023-10-22": {
  "views": 87,
  "downloads": 22
},
"2023-10-23": {
  "views": 79,
  "downloads": 18
},
"2023-10-24": {
  "views": 68,
  "downloads": 15
},
}
```

```

    "2023-10-25": {
      "views": 74,
      "downloads": 17
    },
    "2023-10-26": {
      "views": 82,
      "downloads": 19
    },
    "2023-10-27": {
      "views": 91,
      "downloads": 23
    },
    "2023-10-28": {
      "views": 95,
      "downloads": 26
    },
    "2023-10-29": {
      "views": 88,
      "downloads": 24
    },
    "2023-10-30": {
      "views": 77,
      "downloads": 21
    }
  }
}

```

Объект DatasetGraphs:

```

"DatasetGraphs": {
  "_id": "b2c3nmn5-f6v7-8g10-hle2-j314l5m5n7o8"
  "graphs": [
    {
      "name": "Январь",
      "data": BinData(0,
        "iVBORw0KGgoAAAANSUHEUgAAABKAAAAZCAYAAADE6YVjAAABhWLDQ1BJQ0MgcHJvZmlsZQAAKJF9kT1
        Iw1AUhU9TPaIVBzuIOGSoThZERRY1CkWoEGqFVh1MXvoHTRqSFBdHwbXg4M9i1cHFWVcHV0EQ/
        ABxdHJSdJES/5cUWsR4cNyPd/
        ced+8AoVFmntU1Dmi6baaTCTGbWxVDrwiieEMSIwujiozAZn+p4u4eH7fi6N40r9zH+hz9JYFLADyR6Z
        ZdN94iJu2Ea8TT2xmKXPYDr08kLcS7xLPN9qj85eEwwl+Jt4Tt9nKQk8T1yJaa3HSNiWfEEyVWlPuZFv
      )
    }
  ]
}

```



```

G8x1zVWmD0y/4sijTS/
2RZjrTQ0YinykGaxAqVVQYlGIjVQhI2YrTrpFhI0bnUw5/5IdcrkKcuQqgpBBHFVn7N/
7uHq1s5CfcPig00D1E5jGAhw4Dg07NY9dw23r2B/tvUwv5F6NZDYD6S7yfy3huA/
wt0d3d93W3XfH+U6fA0JvYLB16yqQYwCA3c3g/3d3d3b+1Wt/
zPjfw6XoDnK3QWHWbqAAAAA1wSFlzAAAUiWAALiMBeKU/
dgAAAAAd0SU1FB+cKEQ4WIg0A7L8AAAAZdEVYdENvbW1lbnQAQ3JLYXRlZCB3aXRoIEEdJTVBXgQ4XAAAA
SElEQVRIx2NgGAXDFmzatMmKAQ38//+f4f///wzY5P7//8/
AyMjIwMjIyIBNDpMcLjVMcpjksMkx4pJjxCWHrgaXHCM+OQAKqRcxz2xw4QAAAABJRU5ErkJggg==")
    }
  ]
}

```

Объект User:

```

"User": {
  "_id": "c3d4e5f6-g7h8-9123-i4j5-k6l7m8n9o0p1",
  "username": "John Sales",
  "login": "john.sales@company.com",
  "password": "pa$$word123",
  "status": 1,
  "createdDatasetsCount": 17,
  "accountCreationDate": ISODate("2022-03-15T08:12:33Z"),
  "lastAccountModificationDate": ISODate("2023-10-28T14:45:21Z")
}

```

3.a.IV. Запросы к модели, с помощью которых реализуются сценарии использования

Добавление пользователя

```

db.UserCollection.insertOne({
  _id: UUID(),
  username: "username",
  login: "login",
  password: "password",
  status: 0,
  createdDatasetsCount: 0,
  accountCreationDate: new Date(),
  lastAccountModificationDate: new Date()
})

```

- Запросов: 1
 - Задействованные коллекции: UserCollection
-

Получение пароля пользователя по его логину

```
db.UserCollection.findOne(  
  { login: "login" },  
  { password: 1 }  
)
```

- Запросов: 1
 - Задействованные коллекции: UserCollection
-

Получение данных пользователя

```
db.UserCollection.findOne({  
  _id: "uuid"  
})
```

- Запросов: 1
 - Задействованные коллекции: UserCollection
-

Получение краткой информации о датасетах для отображения карточек на главной странице

```
db.DatasetInfoCollection.find(  
  { },  
  { _id: 1, name: 1, description: 1, size: 1 }  
)
```

- Запросов: 1
 - Задействованные коллекции: DatasetInfoCollection
-

Получение всех данных об одном датасете

```
db.DatasetInfoCollection.findOne({  
  _id: "uuid"  
})
```

```
db.DatasetGraphsCollection.findOne({
  _id: "uuid"
})
```

```
db.DatasetActivityCollection.findOne({
  _id: "uuid"
})
```

- Запросов: 3
 - Задействованные коллекции: DatasetInfoCollection, DatasetGraphsCollection, DatasetActivityCollection
-

Добавление датасета

```
db.DatasetInfoCollection.insertOne({
  _id: UUID(),
  name: "name",
  description: "description",
  creationDate: new Date(),
  author: "author",
  rowCount: 12500,
  columnCount: 24,
  size: 5242880,
  path: "path",
  lastVersionNumber: 1,
  lastModifiedDate: new Date(),
  lastModifiedBy: "author"
})
```

```
db.DatasetActivityCollection.insertOne({
  _id: UUID(),
  statistics: {
    "2023-10-01": {
      "views": 42,
      "downloads": 7
    },
    "2023-10-02": {
      "views": 38,
      "downloads": 5
    }
  }
})
```

```

    }
  })

db.DatasetGraphsCollection.insertOne({
  _id: UUID(),
  graphs: [
    {
      name: "name",
      data: BinData(0,
        "iVBORw0KGgoAAAANSUHEUgAAABKAAAAZCAYAAADE6YVjAAABhWLDQ1BJQ0MgCHJvZmlsZQAACKJF9kT1
        Iw1AUhU9TpaIVBzuIOGSoThZERRY1CkWoEGqFVh1MXvoHTRqSFBdHwbXg4M9i1cHFWVcHV0EQ/
        ABxdHJSdJES/5cUWsR4cNyPd/
        ce8AoVFlmtU1Dmi6baaTCTGbWxVDrwiieEMSIwujiozAZn+p4u4eH7fi6N40r9zH+hz9JYFLADyR6ZZd
        N94iJu2Ea8TTxmKXPyDr08kLcS7xLPN9qj85eEwwl+Jt4Tt9nKQk8T1yJaa3HSNiWfEEyVwLPuZFvG8x
        1zVwmD0y/4sijTS/
        2RZjrTQ0YinykGaxAqVVQYlGIjVQhI2YrTrpFhI0bnUw5/5IdcrkKcuQqgpBBHFVn7N/
        7uHq1s5CfcPig00D1E5jGAhW4Dg07NY9dW23r2B/tvUwv5F6NZDYD6S7yfyY3huA/
        wt0d3d93W3XFH+U6fA0JvYLB16yqQYwCA3c3g/3d3d3b1Wt/
        zPjfw6XoDnK3QWHWbqAAAAA1wSF1zAAAUiWAALiMBekU/
        dgAAAAAd0SU1FB+cKEQ4WIg0A7L8AAAAZdEVYdENvbW1lbnQAQ3JlYXRlZCB3aXRoIEEdJTVBXGQ4XAAAA
        SE1EQVRlX2NgGAXDFmzatMmKAQ+f4f///wzY5P7//8/
        AyMjIwMjIyIBNDpMcLjVMcpjksMkx4pJjxCWHrgaXHCM+0QqRcxz2xw4QAAAABJRu5ErkJggg==")
      }
    ]
  })

db.UserCollection.updateOne(
  { _id: "user_uuid" },
  { $set: {
    createdDatasetsCount: old_count + 1
  }
  }
)

```

- Запросов: 4
- Задействованные коллекции: DatasetInfoCollection, DatasetGraphsCollection, DatasetActivityCollection, UserCollection

Изменение датасета

```
db.DatasetInfoCollection.updateOne(
  { _id: "uuid" },
  { $set: {
    name: "new_name",
    description: "new_description",
    rowCount: new_row_count,
    columnCount: new_column_count,
    size: new_size,
    lastVersionNumber: new_version,
    lastModifiedDate: new Date(),
    lastModifiedBy: "modified_by_name"
  }
})
```

```
db.DatasetGraphsCollection.updateOne(
  { _id: "uuid" },
  { $set: { graphs: new_graphs_array } }
)
```

- Запросов: 2
 - Задействованные коллекции: DatasetInfoCollection, DatasetGraphsCollection
-

Редактирование пользователя

```
db.UserCollection.updateOne(
  { _id: "uuid" },
  { $set: {
    username: "new_username",
    login: "new_login",
    password: "new_password",
    lastAccountModificationDate: new Date()
  }
})
```

- Запросов: 1

- Задействованные коллекции: UserCollection
-

Удаление датасета

```
db.DatasetInfoCollection.deleteOne({  
  _id: "uuid"  
})
```

```
db.DatasetGraphsCollection.deleteOne({  
  _id: "uuid"  
})
```

```
db.DatasetActivityCollection.deleteOne({  
  _id: "uuid"  
})
```

- Запросов: 3

Задействованные коллекции: DatasetInfoCollection, DatasetGraphsCollection, DatasetActivityCollection

3.b. Реляционная модель

В базе данных созданы четыре таблицы:

- DataSet - таблица, содержащая метаданные датасета;
- ColumnInfo - таблица, содержащая графики для визуализации;
- Activity - таблица, содержащая статистику просмотров и скачиваний за последние 30 дней;
- User - таблица, содержащая данные о пользователях.

Каждая из таблиц описана более подробно ниже.

3.б.І. Графическое представление

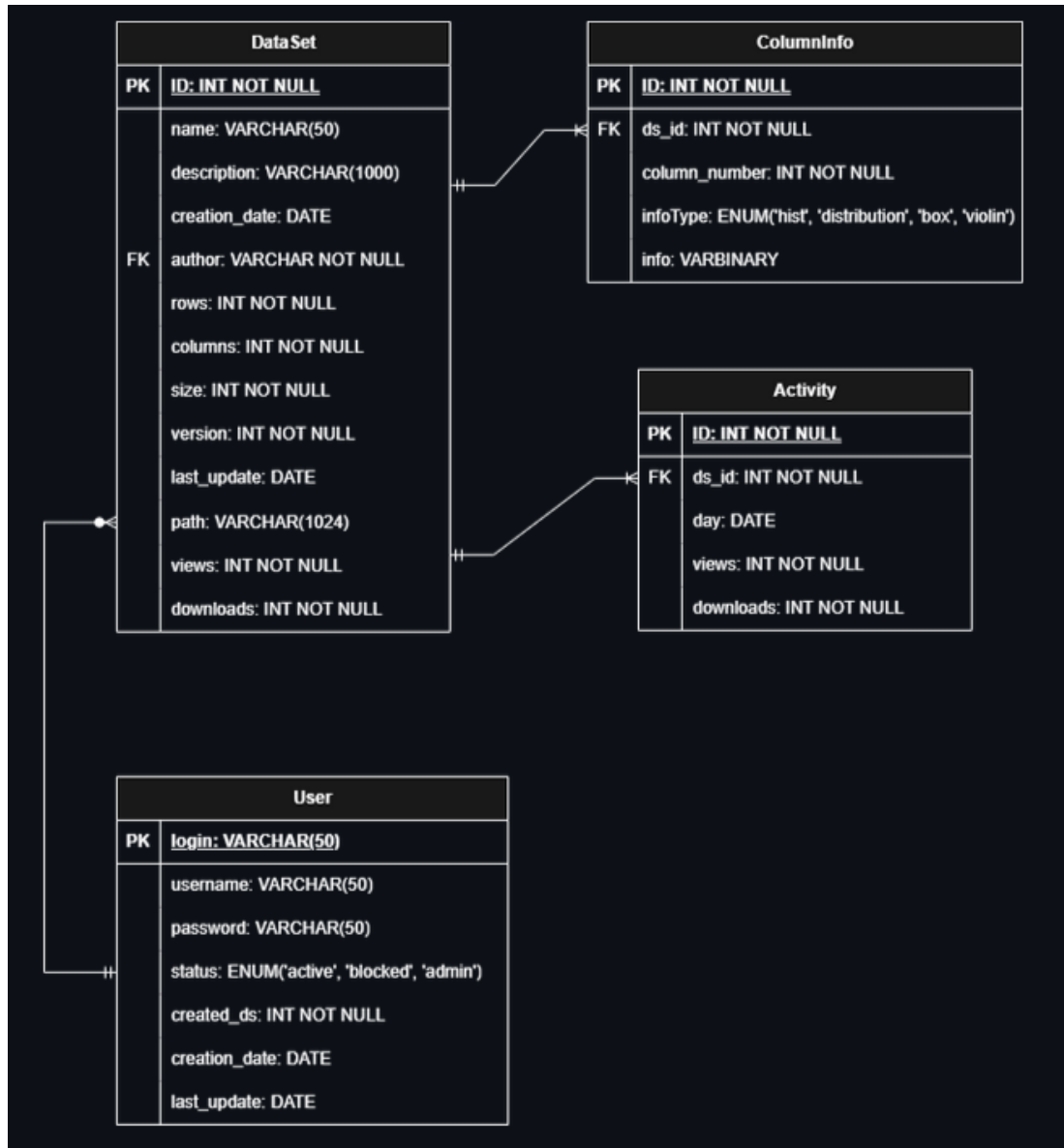


Рисунок 2: ER-диаграмма

3.б.ІІ. Описание назначений коллекций, типов данных и сущностей

Таблица DataSet

Данная таблица хранит общую информацию о датасете, такую как название, описание, размер и проч.

Название поля	Тип данных	Объем, байт
id	Целочисленный	4
name	Строка до 50 символов	50
description	Строка до 1000 символов	1000

Название поля	Тип данных	Объем, байт
creation_date	Дата	8
author	Строка до 50 символов	50
rows	Целочисленный	4
columns	Целочисленный	4
size	Целочисленный	4
version	Целочисленный	4
last_update	Дата	8
path	Строка до 1024 символов	1024
views	Целочисленный	4
downloads	Целочисленный	4

Таблица ColumnInfo

Данная Таблица хранит графики для визуализации информации о признаках набора данных.

Название поля	Тип данных	Объем, байт
id	Целочисленный	4
ds_id	Целочисленный	4
column_number	Целочисленный	4
infoType	Перечисляемый	4
info	Файл	

Так как файлы хранятся как бинарники их размер может варьироваться.

Таблица Activity

Данная таблица хранит статистику просмотров и скачиваний за последние 30 дней.

Название поля	Тип данных	Объем, байт
id	Целочисленный	4
ds_id	Целочисленный	4
day	Дата	8
views	Целочисленный	4
downloads	Целочисленный	4

Таблица User

Данная таблица хранит информацию о пользователях, такую как имя, логин, пароль, статус и проч.

Название поля	Тип данных	Объем, байт
login	Строка до 50 символов	50
username	Строка до 50 символов	50
password	Строка до 50 символов	50
status	Перечисляемый	4
created_ds	Целочисленный	4
creation_Date	Дата	8
last_update	Дата	8

3.b.III. Оценка удельного объема информации, хранимой в модели

При добавлении нового датасета создаются записи в DataSet и Activity, а также для каждого признака создаются записи в ColumnInfo, для каждого числового признака 3 записи, а для каждого категориального - 1.

Таким образом, при добавлении одного датасета в БД необходимо выделить:

$$2168 + 30 \cdot 24 + \left(n \cdot 16 + \sum_{i=1}^n \sum_{j=1}^3 p_{ij} \right) + \left(c \cdot 16 + \sum_{i=1}^c s_i \right)$$

Где n - количество числовых признаков, c - категориальных, p_{ij} - размер файла j -го графика i -го числового признака, s_i - размер файла графика i -го категориального признака.

При добавлении нового пользователя создается запись в User.

Таким образом, при добавлении одного пользователя в БД необходимо выделить 174 байта.

А значит весь объем информации в БД занимает в байтах

$$2888 \cdot N + \sum_{k=1}^N \left(\left[n_k \cdot 16 + \sum_{i=1}^{n_k} \sum_{j=1}^3 p_{kij} \right] + \left[c_k \cdot 16 + \sum_{i=1}^{c_k} s_{ki} \right] \right) + 174 \cdot M,$$

где N - количество датасетов в системе, а M - количество пользователей в системе, n_k - количество числовых признаков k -го датасета, c_k - категориальных k -го датасета, p_{kij} - размер файла j -го графика i -го числового признака k -го датасета, s_{ki} - размер файла графика i -го категориального признака k -го датасета.

Предположим, что на один датасет приходится 10 пользователей (по сравнению с количеством датасетов количество пользователей в разы больше, так как большинство пользователей не загружают новые датасеты, а пользуются уже существующими).

Соответственно, $M = 10 \cdot N$

Предположим, что в среднем на один датасет приходится 5 числовых и 2 категориальных признака, а каждый файл весит 100КБ, т.е

$$n_k = 5, c_k = 2, \forall k;$$

$$p_{kij} = s_{vu} = 102400, \forall k, i, j, v, u;$$

Тогда получаем объем информации в зависимости от количества датасетов:

$$V(N) = 2888 \cdot N + N(5 \cdot 16 + 15 \cdot 102400 + 2 \cdot 16 + 2 \cdot 102400) + 174 \cdot (10 \cdot N) = 1,745,540 N$$

Избыточность данных

Оценка чистого объема данных таблицы DataSet:

Название поля	Объем, байт
name	50
description	1000
creation_date	8
rows	4
columns	4
size	4
	34

Название поля	Объем, байт
version	4
last_update	8
path	1024
views	4
downloads	4

Оценка чистого объема данных таблицы ColumnInfo

Название поля	Объем, байт
column_number	4
infoType	4
info	

Оценка чистого объема данных таблицы Activity

Название поля	Объем, байт
day	8
views	4
downloads	4

Таким образом, при добавлении одного датасета в БД необходимо выделить в байтах

$$2114 + 30 \cdot 16 + \left(n \cdot 8 + \sum_{i=1}^n \sum_{j=1}^3 p_{ij} \right) + \left(c \cdot 8 + \sum_{i=1}^c s_i \right)$$

Где n - количество числовых признаков, c - категориальных, p_{ij} - размер файла j -го графика i -го числового признака, s_i - размер файла графика i -го категориального признака.

Оценка чистого объема данных таблицы User

Название поля	Объем, байт
login	50
username	50
password	50

Название поля	Объем, байт
status	4
created_ds	4
creation_Date	8
last_update	8

Таким образом, при добавлении одного пользователя в БД необходимо выделить 174 байта.

А значит весь чистый объем данных в БД составляет

$$2594 \cdot N + \sum_{k=1}^N \left(\left[n_k \cdot 8 + \sum_{i=1}^{n_k} \sum_{j=1}^3 p_{kij} \right] + \left[c_k \cdot 8 + \sum_{i=1}^{c_k} s_{ki} \right] \right) + 174 \cdot M,$$

где N - количество датасетов в системе, а M - количество пользователей в системе.

Воспользовавшись теми же предположениями получим чистый объем от числа датасетов

$$V_c(N) = 1,745,190 \text{ байт}$$

В таком случае, коэффициент избыточности будет равен:

$$C = V(N) / V_c(N) = (1,745,540 \cdot N) / (1,745,190 \cdot N) \approx 1.0002$$

Направление роста модели при увеличении количества объектов каждой сущности.

Из формулы -

$$2888 \cdot N + \sum_{k=1}^N \left(\left[n_k \cdot 16 + \sum_{i=1}^{n_k} \sum_{j=1}^3 p_{kij} \right] + \left[c_k \cdot 16 + \sum_{i=1}^{c_k} s_{ki} \right] \right) + 174 \cdot M$$

видно, что модель растет линейно относительно каждого из параметров.

Примеры данных

Таблица DataSet:

id	name	description	creation_date	author	rows	columns	size	version	last_updated	path	views	downloads
1	Sales Data 2023	This dataset contains information about sales	2023-07-15T09:30:45Z	John Sales	12500	24	52428850		2023-09-20T14:15:22Z	/dataset/sales/dataset.csv	77	21

Таблица ColumnInfo:

id	ds_id	column_number	infoType	info
1	1	1	violin	“bnarescdssd... gfgsdfkkjlfad”
2	1	1	box	“asfhgsvsdrh... dsfefregfdr”
3	1	2	hist	“cmcyrhgjdfig... nvxnghgyjtthgj”

Таблица Activity:

id	ds_id	day	views	downloads
1	1	2023-10-01	38	5
2	1	2023-10-01	42	7
3	1	2023-10-01	55	12

Таблица User:

login	username	password	status	created_ds	creation_Date	last_update
john	John Sales	pa\$\$word123	active	17	2022-03-15T08:12:33Z	2023-10-28T14:45:21Z

3.b.IV. Запросы к модели, с помощью которых реализуются сценарии использования

- Регистрация пользователя:

```
INSERT INTO User(login, username, password, status, created_ds, creation_date, last_update)
VALUES ("james", "James Jameson", "pass", "active", 0, "2024-04-01", "2024-04-01")
```

Количество запросов: 1

Количество задействованных таблиц: 1

- Вход в сервис

```
SELECT login, password FROM User
```

Количество запросов: 1

Количество задействованных таблиц: 1

- Фильтр и сортировка датасетов

```
SELECT * FROM DataSet
WHERE
    name = "%sales%"
    AND size BETWEEN 3024 AND 16996
    AND downloads BETWEEN 70 AND 100
    AND views BETWEEN 200 AND 300
    AND creation_date BETWEEN "2023-01-01" AND "2024-01-01"
ORDER BY views
```

Количество запросов: 1

Количество задействованных таблиц: 1

- Добавление датасета

```
INSERT INTO DataSet(
    name,
    description,
    creation_date,
    author,
```

```

        rows,
        columns,
        size,
        version,
        last_update,
        path,
        views,
        downloads
    )
VALUES (
    "iris",
    "iris dataset",
    "2024-01-01",
    "james",
    300,
    2,
    12000,
    1,
    "2024-01-01",
    "/datasets/iris/iris.csv",
    0,
    0
)

INSERT INTO ColumnInfo(ds_id, column_number, infoType, info)
VALUES
    (2, 1, "distribution", "fadgfsasdhgdsdfsfga...dfsga"),
    (2, 1, "box", "snvxbagsdgdgasdfe...xcbzcb"),
    (2, 1, "violin", "mjghsgddfnxfgheag...xcbcbz"),
    (2, 2, "hist", "ngeradgadsggrargsdf...gasfaf")

INSERT INTO Activity(ds_id, day, views, downloads)
VALUES (2, "2024-01-01", 0, 0)

UPDATE User
SET created_ds = created_ds + 1
WHERE login = "james"

```

Количество запросов: 4

Количество задействованных таблиц: 4

- Просмотр информации датасета

```
SELECT name, description, author, rows, columns, size, creation_date, version,  
last_update, views, downloads FROM DataSet  
WHERE name = "iris"
```

```
SELECT info FROM ColumnInfo  
WHERE  
    ds_id = (SELECT id FROM DataSet WHERE name = "iris") AND (  
        column_number = 1 AND infoType = "box"  
        OR column_number = 2 AND infotype = "hist"  
    )
```

Количество запросов: 3

Количество задействованных таблиц: 2

- Просмотр активности датасета

```
SELECT views, downloads FROM Activity  
WHERE ds_id = (SELECT id FROM DataSet WHERE name = "iris")
```

Количество запросов: 2

Количество задействованных таблиц: 2

- Скачать датасет

```
SELECT path FROM DataSet  
WHERE name = "iris"
```

Количество запросов: 1

Количество задействованных таблиц: 1

- Редактировать датасет

```
UPDATE DataSet  
SET name = "iris2",  
    description = "iris super dataset",  
    version = version + 1  
    last_update = "2024-04-03"  
WHERE name = "iris"
```

```
UPDATE ColumnInfo  
SET info = "asdgafigadadfgasd...adfhds"
```


WHERE

```
ds_id = (SELECT id FROM DataSet WHERE name = "iris2")
AND column_number = 1 AND infoType = "box"
```

UPDATE ColumnInfo

```
SET info = "nsfbsgdfgsdfgsrrea...vcxvd"
```

WHERE

```
ds_id = (SELECT id FROM DataSet WHERE name = "iris2")
AND column_number = 1 AND infoType = "violin"
```

UPDATE ColumnInfo

```
SET info = "baerfdfgasdgrgser...bzxvcb"
```

WHERE

```
ds_id = (SELECT id FROM DataSet WHERE name = "iris2")
AND column_number = 1 AND infoType = "distribution"
```

Количество запросов: $3 \cdot n + c + 1$ (n - число измененных числовых столбцов c)

Количество задействованных таблиц: 2

- Удалить датасет

DELETE FROM ColumnInfo

```
WHERE ds_id = (SELECT id FROM DataSet WHERE name = "iris")
```

DELETE FROM Activity

```
WHERE ds_id = (SELECT id FROM DataSet WHERE name = "iris")
```

DELETE FROM DataSet

```
WHERE name = "iris"
```

Количество запросов: 5

Количество задействованных таблиц: 3

- Просмотр страницы пользователя

```
SELECT login, username, created_ds, creation_date, last_update FROM User
WHERE username = "James Jameson"
```

Количество запросов: 1

Количество задействованных таблиц: 1

- Изменение пользователя

```
UPDATE User
SET
    username = "James",
    password = "ubepass",
    last_update = "2024-04-02"
```

Количество запросов: 1

Количество задействованных таблиц: 1

- Массовый экспорт

```
SELECT name, path FROM DataSet
```

Количество запросов: 1

Количество задействованных таблиц: 1

- Блокировка пользователя

```
UPDATE User
SET
    username = "James",
    status = "blocked"
```

Количество запросов: 1

Количество задействованных таблиц: 1

3.с. Сравнение моделей

3.с.1. Удельный объем информации

Параметр	NoSQL	SQL
Формула	$703 \cdot N + 0.2 \cdot M$	$2888 \cdot N + N \cdot (n_k \cdot 16 + 15 \cdot 102400 + c_k \cdot 16 + c_k \cdot 102400) + 174 \cdot M$
объема, при учёте V(N,M) (кбайт)		
Объём при учёте M=10N (кбайт)	705	1704
Избыточность	1.004	1.0002

Из полученных данных можно сделать вывод о том, что несмотря на меньшую избыточность реляционной модели, нереляционная модель занимает примерно в 2.4 раза меньше места (при учёте что каждый файл весит 100КБ). Это свидетельствует о том, что нереляционная модель более предпочтительна с точки зрения занимаемого места.

3.с.II. Запросы по отдельным юзкейсам

Параметр	NoSQL	SQL
Добавление пользователя	1 вызов <code>insertOne (User)</code>	1 <code>INSERT (User)</code>
Получение пароля пользователя по логину	1 вызов <code>findOne (User)</code>	1 <code>SELECT (User)</code>
Получение данных пользователя	1 вызов <code>findOne (User)</code>	1 <code>SELECT (User)</code>
Добавление датасета	4 вызова (1 <code>insertOne Info</code> , 14 запроса (1 <code>INSERT DataSet</code> , 1 <code>insertOne Graphs</code> , 1 <code>insertOne Activity</code> , 1 <code>updateOne User</code>)	<code>INSERT ColumnInfo</code> , 1 <code>INSERT Activity</code> , 1 <code>UPDATE User</code>)
Редактирование пользователя	1 вызов <code>updateOne (User)</code>	1 <code>UPDATE (User)</code>
Удаление датасета	3 вызова <code>deleteOne (Info, Graphs, Activity)</code>	3 <code>DELETE (ColumnInfo, Activity, DataSet)</code>

Для простых операций с пользователями (добавление, получение данных, редактирование) обе модели требуют одного запроса к одной коллекции/таблице. Сложность соизмерима.

Получение информации о датасете в NoSQL требует трех отдельных запросов по `_id` к трем разным коллекциям. В SQL это также требует запросов к трем таблицам (`DataSet`, `ColumnInfo`, `Activity`), которые могут быть выполнены как отдельными запросами, так и объединены в один `JOIN`. Сложность запросов также соизмерима.

Однако, операции, связанные с изменением и удалением записей выполняются проще в нереляционной модели из-за отсутствия необходимости согласования связанных таблиц.

3.с.III. Вывод - что лучше, SQL или NoSQL модель

В ходе сравнения нереляционной и реляционной моделей были сделаны следующие выводы:

1. Нереляционная модель значительно (~2.4 раза) компактнее в предложенном сценарии, в основном из-за способа хранения массивов графиков и статистики активности.
2. Для базовых операций над пользователями сложность сопоставима. Для операций, затрагивающих весь датасет (добавление, изменение, удаление), NoSQL модель упрощает взаимодействие с данными.

Исходя из всего вышесказанного, следует отдать предпочтение нереляционной модели данных.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

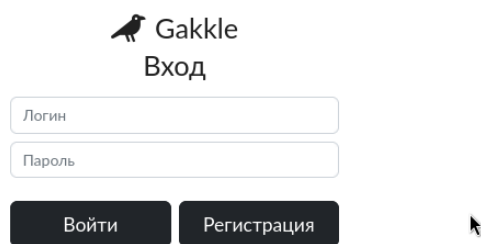
4.a. Краткое описание

Docker Compose файл включает три сервиса: db (MongoDB 8.0.6), mongo-express (веб-админ-панель для MongoDB) и app (Flask-приложение). База данных db настраивается через переменные окружения с правами root, монтирует том *dbdata* для хранения данных и проверяет свою доступность через healthcheck. Сервис mongo-express зависит от работоспособности db, подключается к нему через аутентификацию и предоставляет веб-интерфейс на порту 8081 (локально). App — это Python-приложение, собранное из Dockerfile, которое взаимодействует с MongoDB через переменную *MONGO_URI*, использует том *datasets* для файлов и публикует порт 5000. Все контейнеры объединены мостовой сетью *service_net*, обеспечивающей изолированное взаимодействие. Конфигурация гибко настраивается через переменные окружения (например, логины, пароли, порты).

4.b. Используемые технологии

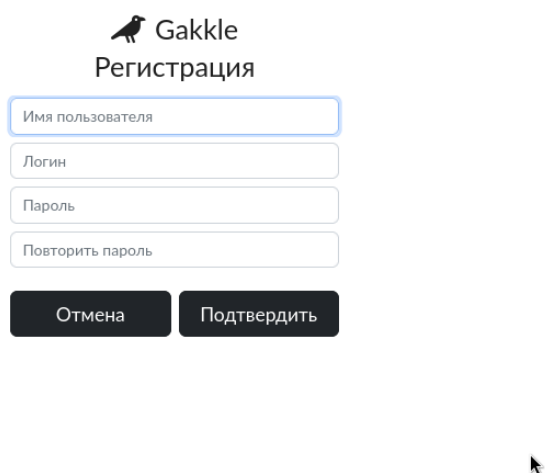
Приложение имеет клиент-серверную архитектуру: веб-интерфейс взаимодействует с backend-частью, написанной на Python с применением фреймворка Flask. Сервер будет обрабатывать запросы от клиента и возвращать требуемые данные. Данные будут получаться сервером из базы данных. База данных MongoDB будет развернута на отдельном узле (контейнере). База данных будет хранить данные пользователей и информацию о датасете (пути до файлов с датасетами, графики, метаданные).

4.с. Снимки экрана приложения




The screenshot shows the login interface for the Gakkle application. At the top, there is a logo consisting of a small bird icon followed by the text "Gakkle". Below the logo is the word "Вход" (Login). There are two input fields: the first is labeled "Логин" (Login) and the second is labeled "Пароль" (Password). Below these fields are two buttons: "Войти" (Login) and "Регистрация" (Registration).

Рисунок 3: Экран входа



The screenshot shows the registration interface for the Gakkle application. At the top, there is a logo consisting of a small bird icon followed by the text "Gakkle". Below the logo is the word "Регистрация" (Registration). There are four input fields: the first is labeled "Имя пользователя" (Username), the second is labeled "Логин" (Login), the third is labeled "Пароль" (Password), and the fourth is labeled "Повторить пароль" (Repeat password). Below these fields are two buttons: "Отмена" (Cancel) and "Подтвердить" (Confirm).

Рисунок 4: Экран регистрации



Главная

Добавить датасет

Administrator

Admin Panel

Выйти

Название *

dataset

Описание


This is dataset

iris.csv

Сохранить

Отмена

Рисунок 5: Экран добавления датасета



Главная

Редактировать dataset

Administrator

Admin Panel

Выйти

Название *

dataset

Описание

This is dataset

Заменить CSV файл

Сохранить

Отмена

Рисунок 6: Экран редактирования датасета

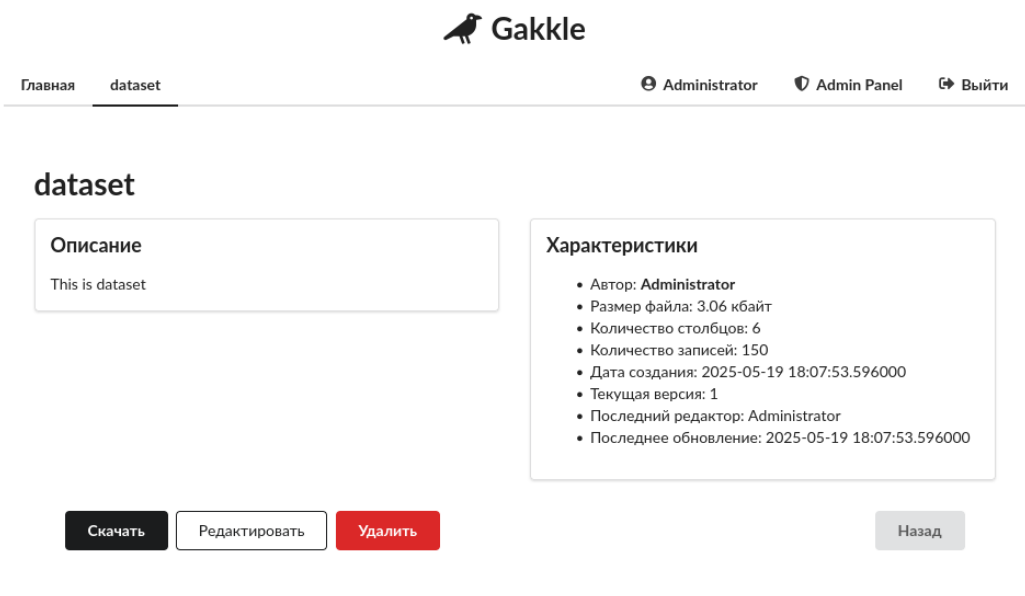


Рисунок 7: Экран датасета

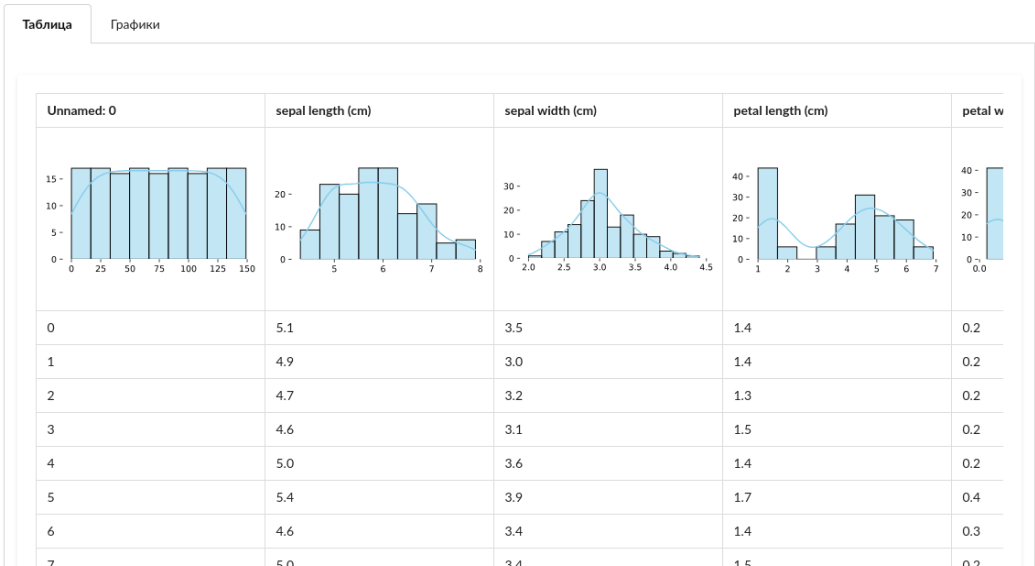


Рисунок 8: Экран датасета (просмотр данных)

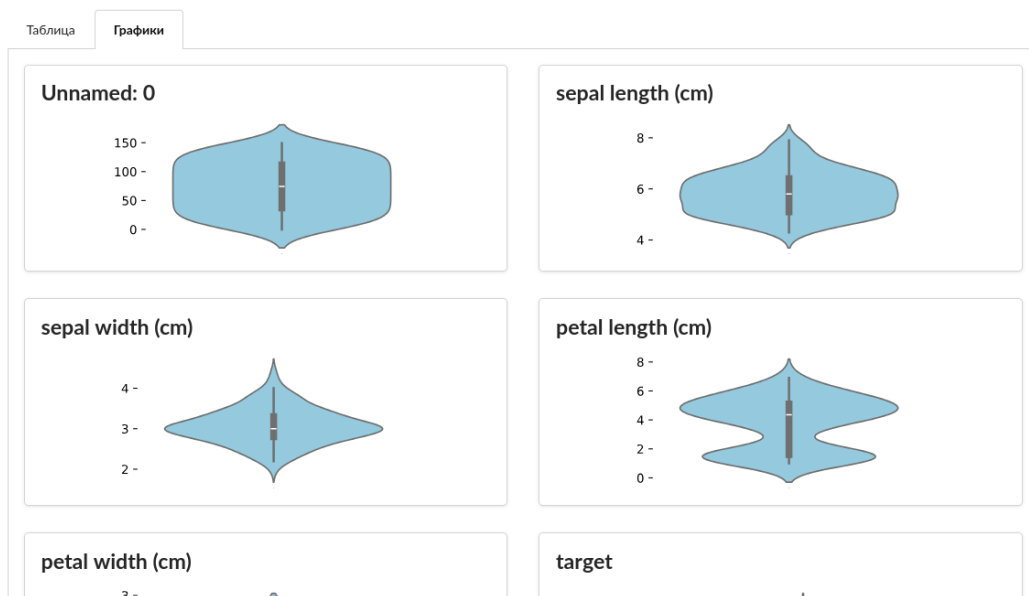


Рисунок 9: Экран датасета (просмотр графиков)

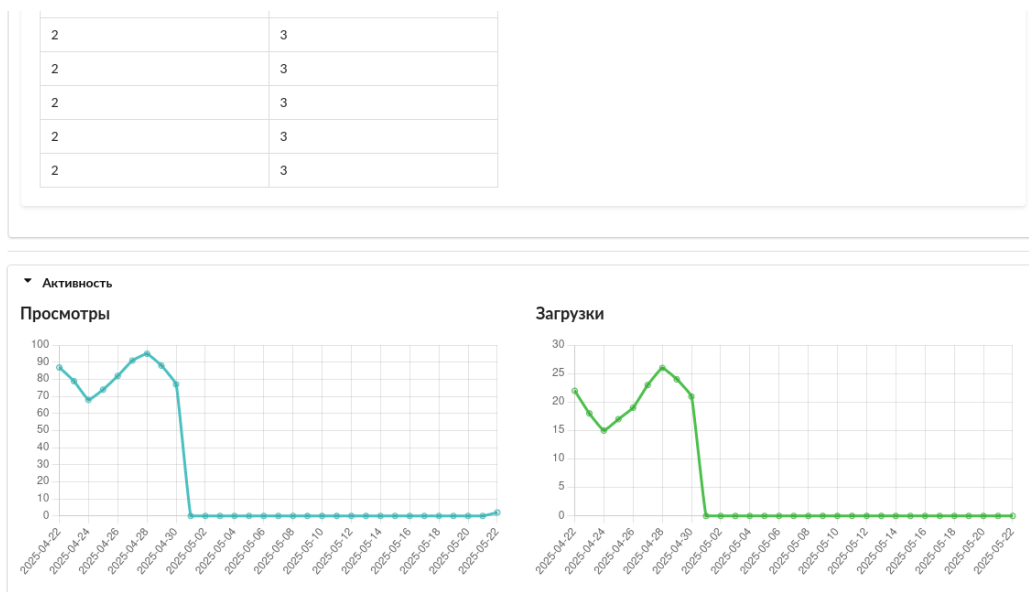



Рисунок 10: Экран датасета (просмотр статистики)



Редактирование профиля

Управление данными вашего аккаунта: Administrator

Имя пользователя

Administrator

Это имя будет отображаться другим пользователям.

Логин

administrator

Изменить пароль

Оставьте поля пустыми, если не хотите менять пароль.

Новый пароль

Минимум 6 символов

Подтвердите новый пароль

Рисунок 11: Экран пользователя

Инструменты

Импорт данных

 Загрузить ZIP файл

 Импорт

Экспорт данных

 Экспорт

Статистика пользователя





-  Дата создания аккаунта
01 May 2025, 00:00:00 UTC
-  Дата последнего изменения аккаунта
01 May 2025, 00:00:00 UTC
-  Количество созданных датасетов
1

Рисунок 12: Экран пользователя
(кнопки импорта/экспорта администратора)



[Главная](#)[Administrator](#)[Admin Panel](#)[Выйти](#)

Пользователи

10

entries per page

Search:

Логин	Имя пользователя	Статус	Создано наборов данных	Дата создания аккаунта	Дата изменения аккаунта	
111	111	Активен	0	2025-05-19 13:40:46.485000	2025-05-19 13:40:46.485000	Заблокировать
administrator	Administrator	Администратор	1	2025-05-01 00:00:00	2025-05-01 00:00:00	Заблокировать
john.sales@company.com	John Sales	Активен	3	2022-03-15 08:12:33	2023-10-28 14:45:21	Заблокировать
kkk	kkk	Активен	0	2025-05-19 12:27:39.924000	2025-05-19 12:27:39.924000	Заблокировать
nnn	lll	Активен	0	2025-05-19 12:28:18.482000	2025-05-19 12:30:41.413000	Заблокировать

Рисунок 13: Экран админ-панели

5. ВЫВОДЫ

5.a. Достигнутые результаты

Разработанный проект представляет собой полнофункциональное веб-приложение для управления датасетами. Успешно внедрены ключевые сценарии использования, включая регистрацию и аутентификацию пользователей, гибкий поиск и сортировку датасетов на главной странице, загрузку и редактирование данных, а также просмотр статистики. Для визуализации данных реализовано отображение датасетов в виде таблиц и графиков, что позволяет пользователям анализировать структуру данных без необходимости сторонних инструментов. Интеграция MongoDB позволила эффективно хранить и обрабатывать данные разной природы, включая графики, метаданные о датасетах, пользователях и их активностях.

Учётная запись администратора предоставляет расширенные возможности: массовый импорт/экспорт данных, управление пользователями (блокировка/разблокировка), а также полный контроль над всеми датасетами.

5.b. Недостатки и пути для улучшения полученного решения

На текущем этапе проект полностью соответствует заявленным требованиям, а все запланированные функции реализованы в полном объёме.

5.c. Будущее развитие решения

Перспективы развития включают внедрение социальных функций (комментарии, рейтинги датасетов), интеграцию с облачными хранилищами для работы с большими объёмами данных, а также добавление поддержки разных форматов датасетов. Дополнительно можно реализовать двухфакторную аутентификацию для пользователей и админов, а также расширить аналитику с использованием ML-моделей для автоматической классификации датасетов.

Проект обладает потенциалом для превращения в универсальную платформу анализа данных, где сочетаются удобство управления, глубокая визуализация и возможности совместной работы.

ПРИЛОЖЕНИЕ

Документация по сборке и разворачиванию приложения

Для начала работы с сервисом необходимо установить Docker — платформу с открытым исходным кодом для автоматизации разработки, доставки и разворачивания приложений. Скачайте Docker Desktop с официального сайта (<https://www.docker.com/products/docker-desktop/>) для вашей операционной системы (Windows, macOS или Linux) и следуйте инструкциям установки. После завершения установки запустите Docker Desktop. Убедитесь, что он работает: в системном трее (на Windows или macOS) или через команду `docker --version` в терминале (на Linux) должна отображаться версия Docker.

Далее откройте терминал (Command Prompt на Windows, Terminal на macOS/Linux) и клонируйте проект. Если вы не знакомы с Git, можно скачать [архив с кодом проекта](#) и распаковать его в удобную папку. Перейдите в папку проекта через терминал, используя команду `cd путь_к_папке_проекта`. Убедитесь, что в этой папке есть файл `docker-compose.yml` и `.env`. Если файла `.env` нет, создайте его вручную и скопируйте в него настройки из описания проекта (значения переменных уже предустановлены, менять их не требуется).

Теперь выполните команду

```
docker compose build --no-cache && docker compose up
```

Это загрузит все необходимые компоненты (базу данных MongoDB, веб-приложение и утилиту для просмотра БД MongoExpress) и запустит их. Процесс может занять несколько минут — дождитесь сообщения в терминале о том, что приложение готово. После этого откройте браузер и перейдите по адресу <http://127.0.0.1:5000/>. Это главная страница сервиса. Если страница не открывается, убедитесь, что Docker Desktop запущен, а в терминале нет ошибок.

Для остановки сервиса нажмите Ctrl+C в терминале, где он запущен, или выполните `docker compose down` в другой вкладке терминала. Чтобы снова запустить сервис, используйте `docker compose up`.

Инструкция для пользователя

Для начала работы откройте в браузере адрес <http://127.0.0.1:5000/>. Вы увидите страницу входа. Если вы используете сервис впервые, в системе уже созданы два тестовых аккаунта:

Администратор:

Логин `administrator`

Пароль `0dhABEwrwwvtZJQw3a0A1IliEVbiQvwd`

Обычный пользователь:

Логин `vasily`

Пароль `AuQ5UIkdEiQ0tpn8`.

Нажмите "Войти в сервис", введите логин и пароль. Если данные верны, вы попадёте на главную страницу. Здесь можно искать датасеты по названию, фильтровать их по размеру, дате добавления или другим параметрам, а также сортировать результаты. Чтобы загрузить новый датасет, нажмите на кнопку со знаком «+» в правом нижнем углу, заполните название, описание и выберите файл. Обратите внимание: сервис автоматически анализирует данные и отображает их в виде таблиц и графиков, но ограничивает показ до 30 строк и 30 столбцов (но это число можно изменить в `.env` файле).

На странице "Просмотр датасета" вы увидите подробную информацию: описание, статистику, графики распределения данных (например, гистограммы для числовых признаков). Здесь же можно скачать датасет, отредактировать его или удалить (если вы его создали или админ). На своей странице профиля

(кнопка в верхнем меню) вы можете изменить имя, пароль или посмотреть статистику по загруженным датасетам.

Для администратора доступны дополнительные функции. На главной странице в верхнем меню появится кнопка "админ панель" (просмотр списка, блокировка аккаунтов). На странице администратора появятся кнопки "Массовый импорт/экспорт данных". Админ может редактировать или удалять любые датасеты, даже созданные другими пользователями.

Важные ограничения:

Сервис отображает только первые 10 000 строк из датасета для построения графиков. Графики строятся только для категориальных признаков с числом уникальных значений не больше 7. В таблицах с данными отображаются первые 30 строк и столбцов датасета. Все данные ограничения можно поменять в .env файле в корне проекта.

Если возникли проблемы, проверьте:

Работает ли Docker Desktop.

Открываете ли вы адрес <http://127.0.0.1:5000/> (не localhost!).

Правильно ли скопированы логин и пароль (регистр символов важен).

Для продвинутых пользователей: базу данных можно просмотреть через Mongo Express по адресу <http://127.0.0.1:8081/> (логин root, пароль pass). Это полезно для отладки, но не требуется для обычной работы.

Теперь вы готовы использовать сервис! Если что-то пошло не так, перезапустите контейнеры через `docker compose down` && `docker compose up` и повторите действия, описанные выше.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Репозиторий с кодом проекта: <https://github.com/moevm/nosql1h25-mldata>
2. Вики-страница репозитория с макетом, сценариями использования и описанием модели данных: <https://github.com/moevm/nosql1h25-mldata/wiki>
3. Документация MongoDB: <https://docs.mongodb.com/manual/>
4. Документация Flask: <https://flask.palletsprojects.com/en/stable/>