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

КУРСОВОЙ ПРОЕКТ

по дисциплине «Введение в нереляционные базы данных»

Тема: Разработка приложения для организации электронных очередей в университете

Студент гр. 8382 _____ Облизанов А.Д.

Студентка гр. 8382 _____ Ивлева О.А.

Студент гр. 8382 _____ Гордиенко А.М.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2021

ЗАДАНИЕ

Студенты Облизанов А.Д., Ивлева О.А., Гордиенко А.М.

Группа 8382

Тема проекта: Разработка приложения для организации электронных очередей в университете.

Исходные данные:

Необходимо реализовать приложение для организации электронных очередей в университете с использованием СУБД 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:

Студент гр. 8382	_____	Облизанов А.Д.
Студентка гр. 8382	_____	Ивлева О.А.
Студент гр. 8382	_____	Гордиенко А.М.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В данном проекте было разработано веб-приложение для организации и управления электронными очередями, которые могут быть использованы в образовательном процессе, с использованием нереляционной СУБД MongoDB. Был разработан макет пользовательского интерфейса, была разработана и проанализирована модель данных, проведено сравнение с решением на реляционной СУБД по эффективности хранения данных, количеству запросов к БД. В приложении реализован сбор статистики о пользователях и анализ статистики об очередях. Серверная часть веб-приложения реализована на фреймворке Flask, клиентская часть – на фреймворке React. Приложение может быть развернуто в виде Docker-контейнера.

In this project, a web application for the organization and management of electronic queues, which can be used in the educational process, was developed using a non-relational MongoDB DBMS. A layout of the user interface was developed, a data model was developed and analyzed, a comparison was made with a solution on a relational DBMS in terms of data storage efficiency, the number of database requests. The application implements the collection of statistics about users and the analysis of statistics about queues. The server part of the web application is implemented on the Flask framework, the client part is implemented on the React framework. The application can be deployed as a Docker container.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современное время вводится все больше электронных средств для ведения образовательного процесса: личные кабинеты, онлайн-курсы и т.п., однако существует мало решений, позволяющих организовать электронную очередь для проведения занятий по сдаче работ, проведению консультаций и бесед со студентами. Стоит отметить, что электронные очереди активно применяются в сферах здравоохранения, государственных услуг, в банках.

Задачи веб-приложения:

- Предоставление преподавателям интерфейса для организации электронные очереди для вышеуказанных учебных процессов и контроля проведения занятия по очереди, планирования систематического проведения очередей.
- Предоставление студентам интерфейса для просмотра доступных очередей, записи в очередь, комментирования очереди, просмотра статистики по очередям определенного типа.
- Предоставление администраторам интерфейса для контроля пользователей, просмотра статистики действий пользователя, архива записей, бэкапа и восстановления базы данных.

Для решения данных задач было разработано веб-приложение, серверная часть которого реализована на Flask, клиентская часть – на фреймворке React. В качестве СУБД была использована MongoDB.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

- Разработанное веб-приложение должно использовать нереляционную СУБД MongoDB
- Должны быть реализованы экспорт и импорт БД
- Приложение должно предоставлять статистику по очередям (количество записей, продолжительность)
- Приложение должно иметь возможность разворачивание в виде Docker-контейнера

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет пользовательского интерфейса

Общий вид макета пользовательского интерфейса представлен на рис. 1.

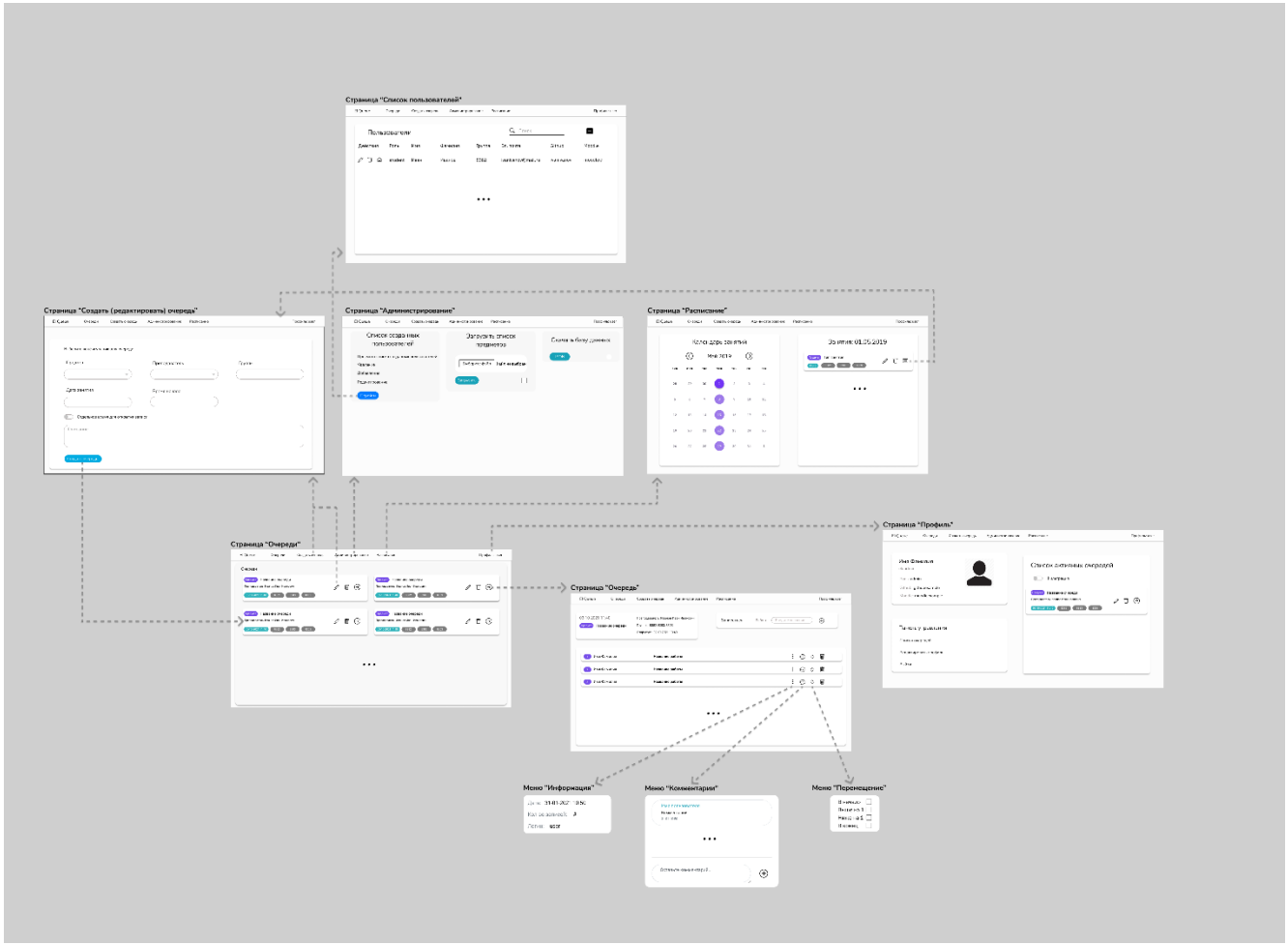

Рисунок 1 – Макет UI, общий вид

Макет главной страницы – списка очередей, представлена на рис. 2.

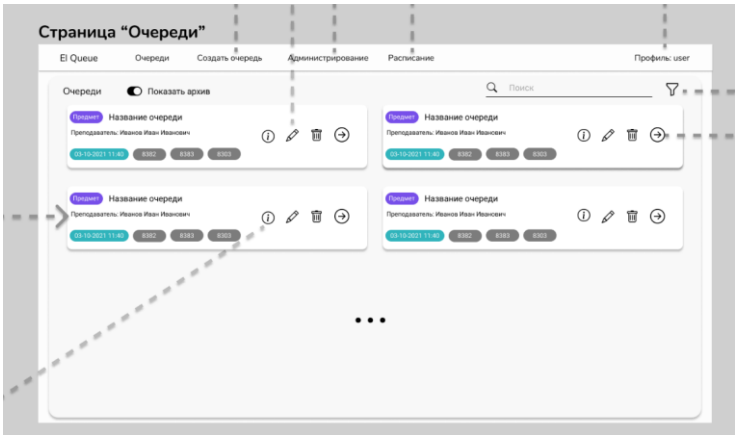

Рисунок 2 – Макет главной страницы

2.2. Сценарии использования для задачи

Сценарии пользователя – студента.

Сценарий “Запись в очередь”

- Студент переходит на страницу “Очереди”
- Студент находит в списке интересующую очереди и нажимает на кнопку со стрелочкой (“перейти”)
- Если запись в очередь еще не доступна, студент может увидеть дату открытия очереди
- Если запись в очередь доступна, в панели “Записаться” студент:
  - Вводит название работы в поле ввода “Работа”
  - Нажимает на кнопку с плюсом (“создать”)
  - Подтверждает действие в всплывающем окне

Сценарий “Изменение профиля”

- Студент переходит на страницу “Профиль”
- В панели управления студент нажимает на кнопку “Редактировать профиль”
- На открывшейся странице студент:
  - Вводит новые данные в поля ввода
  - Нажимает на кнопку “Сохранить”

Сценарии пользователя – преподавателя.

Сценарий “Создание очереди”

- Преподаватель переходит на страницу “Создать очередь”
- В панели “Выберите конфигурацию для очереди” преподаватель:
  - Выбирает предмет занятия из выпадающего списка
  - Выбирает группы студентов из выпадающего списка (или указывает через запятую)
  - Выбирает дату и время начала занятия (datepicker, timepicker)
  - Вводит описание очереди

  - Если время открытия записи отличается от времени начала занятия:
    - переключает флажок “Отдельное время для открытия записи”
    - выбирает дату и время начала записи
- В панели “Предпросмотр” преподаватель может увидеть, как студентам будет представляться очередь в списке
- Для создания очереди преподаватель нажимает кнопку “Создать очередь”

Сценарий “Удаление очереди”

- Преподаватель переходит на страницу “Очереди”
- Преподаватель находит в списке интересующую очереди и нажимает на кнопку с ведром (“удалить”)
- Преподаватель подтверждает действие во всплывающем окне, нажав кнопку “Удалить очередь”

Сценарий “Изменение очереди”

- Преподаватель переход на страницу “Очереди”
- Преподаватель находит в списке интересующую очереди и нажимает на кнопку с карандашом (“изменить”)
- На открывшейся странице преподаватель меняет данные об очереди аналогично созданию очереди
- Для сохранения изменений преподаватель нажимает кнопку “Сохранить” ### Сценарий “Управление записями в очереди”
- Преподаватель переходит на страницу “Очереди”
- Преподаватель находит в списке интересующую очереди и нажимает на кнопку со стрелочкой (“перейти”)

Подсценарий “Комментирование записи студента”

- Преподаватель находит в списке интересующую запись и нажимает на кнопку с сообщением (“комментировать”)
- В открывшемся всплывающем окне преподаватель:

- Вводит комментарий в поле “Оставьте комментарий…”
- При необходимости может изменить размер поля для комментария (при большой его длине)
- Нажимает на кнопку со стрелочкой (“отправить”)

• Преподаватель нажимает в любую часть экрана кроме всплывающего окна для его закрытия

Подсценарий “Перемещение записи”

• Преподаватель в списке интересующую запись, наводит на нее мышку и зажимает ЛКМ
• Преподаватель мышкой переносит запись на необходимую позицию и отжимает ЛКМ

Подсценарий “Удаление записи”

• Преподаватель находит в списке интересующую запись и нажимает на кнопку с ведром (“удалить”)
• Преподаватель действие во всплывающем окне, нажав кнопку “Удалить запись”

Подсценарий “Просмотр дополнительной информации о записи и студенте”

• Преподаватель находит в списке интересующую запись и нажимает на кнопку с тремя точками
• Преподаватель просматривает информацию во всплывающем рядом с кнопкой окне
• Преподаватель нажимает в любую часть экрана кроме всплывающего окна для его закрытия

Сценарии пользователя – администратора.

Сценарий “Экспорт данных”

- Администратор переходит на страницу “Администрирование”

Подсценарий “Экспорт всей базы данных”

- В панели “Скачать базу данных” администратор нажимает кнопку “Экспорт в JSON”

Подсценарий “Экспорт данных о пользователях”

- В панели “Скачать базу данных” администратор нажимает кнопку “Данные о пользователях”
- В открывшемся окне в таблице пользователей администратор выбирает с помощью флажков пользователей, данные о которых хочет скачать
- Администратор нажимает “Экспорт в JSON”

Сценарий “Загрузка списка предметов”

- Администратор переход на страницу “Администрирование”
- В панели “Загрузить список предметов” администратор нажимает на кнопку “Выбор файла”
- В открывшемся проводнике администратор выбирает файл со списоком предметов
- После закрытия проводника администратор нажимает на кнопку “Загрузить” в панели “Загрузить список предметов”

2.3. Вывод

В результате построения макета и определения сценариев использования было выявлено, что основные операции при взаимодействии пользователей с БД будут операциями записи. Инструмент электронных очередей подразумевает активное взаимодействие с ними: запись, продвижение в очереди, добавление комментариев и т.п.

3. МОДЕЛЬ ДАННЫХ

3.1. Структура нереляционной СУБД

Для разработки БД был использован инструмент mognoengine - средство сопоставления объектов и документов для Python.

Формат описания

```
Название_коллекции {
   поле = тип (свойства)
   поле = тип (свойства) // комментарий
   ...
}
```

Особенности:

- ReferenceField - ссылка на документ с помощью DBRef (по ObjectId)
- EmbeddedDocumentField - "вложенный" документ (не ссылка)

Коллекции НСУБД El Queue

1. Коллекция для хранения данных о пользователях

```
User {
   _id: ObjectId // ID, генерируется автоматически
   login = StringField(required=True) // логин пользователя
   password = StringField(max_length=50) // пароль пользователя (хранится в
зашифрованном виде)
   email = StringField(max_length=50) // почтовый ящик пользователя
   name = StringField(required=True, max_length=50) // имя пользователя
   surname = StringField(required=True, max_length=50) // фамилия пользователя
   patronymic = StringField(max_length=50) // отчество пользователя (если есть)
   role = IntField(required=True) // роль пользователя, соответствие чисел и
ролей задается на стороне сервера
   githubID = StringField(max_length=50) // ID в GitHub
   githubLogin = StringField(max_length=50) // Логин в GitHub
   moodleID = StringField(max_length=50) // ID в Moodle
   moodleLogin = StringField(max_length=50) // Логин в Moodle
   group = StringField(max_length=50) // группа - строкой, на случай, если в
ВУЗе предусмотрено использование букв в названии групп
   registerOn = DateField(required=True) // хранение даты регистрации
пользователя
   telemetry = Array(EmbeddedDocument(
      Telemetry {
         timestamp = Timestamp(required=True) // временная отметка действия
         description = String(required=True) // описание действия
         actionType = IntField(required=True) // код действия (соответствие
определяется сервером)
         relatedQueue = ReferenceField(Queue) // не обязательный параметр,
```

```
сохраняется, если пользователь производил действия внутри очереди
      })
   ) // массив телеметрии о действиях пользователя
}
```

2. Коллекция для хранения занятий в расписании

```
Class {
   _id = ObjectId // ID (генерируется автоматически)
   disciplineName = StringField(required=True) // полное название дисциплины
   datetime = StringField(required=True) // Дата проведения занятия (первое
занятие)
   repeatTime = IntField() // Частота повтора занятия (в днях)
   author = ReferenceField(User) // ссылка на документ пользователя-создателя
занятия
   description = StringField() // дополнительная информация о занятии
   type = StringField(required=True) // тип занятия (практика, лекция и т.п.)
   groups = ListField(StringField(), required=True) // список групп, к которым
относится занятие
}
```

3. Коллекция для хранения данных об очередях

```
Queue {
   _id: ObjectId // ID, генерируется автоматически
   discipline = StringField(required=True) // дисциплина, по которой проводится
очередь
   classRef = ReferenceField(Class) // ссылка на документ занятия для случаев,
когда очередь привязана к занятию (не обязателен!)
   students = DictField() // словарь для хранения количества записей каждого
студента в очередь
   groups = ListField(StringField()) // список групп, к которым относится
очередь (не обязателен, так как информация может граниться в занятии)
   date = StringField(required=True) // дата проведения очереди
   time = StringField(required=True) // время проведения очереди
   author = ReferenceField(User) // ссылка на документ пользователя-создателя
очереди
   teacher = StringField() // для случаев, когда необходимо указать нескольких
организаторов
   description = StringField(required=True) // описание очереди (назначение)
   customStart = BooleanField() // флаг, обозначающий отдельное время для
открытия записи в очередь
   startDate = StringField(required=True) // дата начала записи в очередь
   startTime = StringField(required=True) // время начала записи в очередь
   records = ListField(EmbeddedDocumentField(
      Record {
         id: ObjectId // ID записи
         student = ReferenceField(User) // ссылка на пользователя-владельца
записи
         index = IntField() // индекс записи (позиция)
```

```
         date = StringField() // дата записи
         task = StringField() // описание задания (название)
         comments = ListField(EmbeddedDocumentField(
            Comment {
               id = ObjectIdField(default=ObjectId) // ID комментария
               author = ReferenceField(User) // ссылка на пользователя-автора
комментария
               date = StringField() // дата создания комментария
               text = StringField() // текст комментария
            }
         )) // массив комментариев к записи
      }
   )) // массив записей в очередь
   archive = ListField(EmbeddedDocumentField(Record)) // массив архивированных
записей в очередь (документ Record уже описан выше)
   archived = BooleanField(default=False) // флаг, обозначающий статус очереди:
false - активна, true - в архиве
}
```

Примечание: многие очереди могут быть связаны с занятием в расписании, благодаря чему можно было бы уменьшить количество полей для очереди. Однако учитывая то, что очередь может быть создана вне занятия (консультация, дистанционная сдача не во время занятий и т.п.), документ очереди создан таким образом, чтобы его можно было использовать без привязки к занятию.

4. Коллекция для хранения данных о дисциплинах

```
Discipline {
   _id = ObjectId // ID (генерируется автоматически)
   name = StringField(required=True) // полное название дисциплины
   short = StringField() // сокращенное название дисциплины
}
```

Схема коллекций НСУБД

Схема представлена на рис. 2.

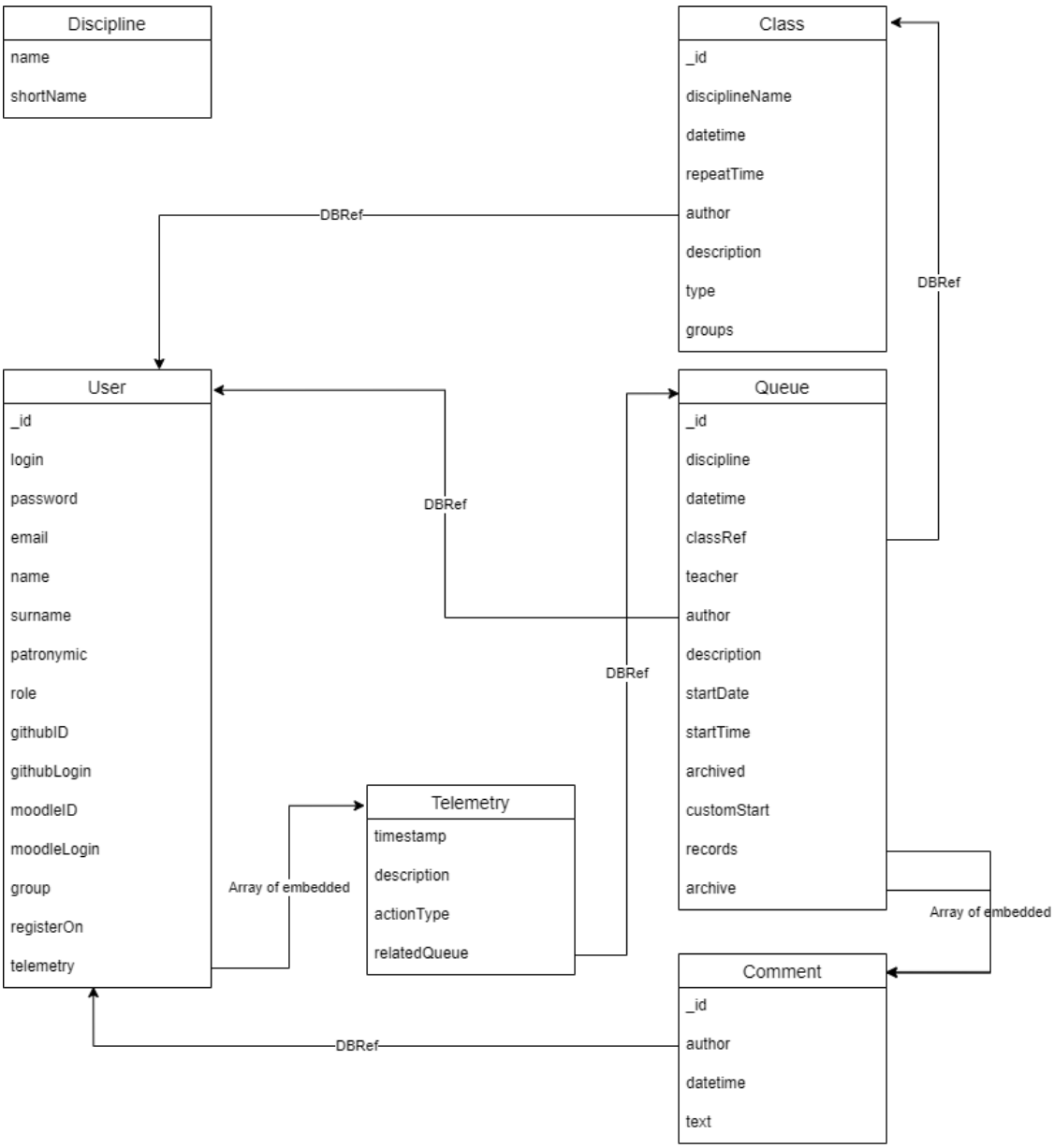

Рисунок 2 – Схема данных MongoDB

Оценка удельного объема информации, хранимой в модели

Размеры типов данных

- ObjectID - 12 байт
- Int - 4 байт
- String - 2*N байт (пусть средняя длина строки - N = 16 символов)
- DBRef ~ ObjectID
- Boolean - 2 байт
- Date - 8 байт

Размеры документов в коллекциях (и вложенных документов)

- size(**Telemetry**): 2 x ObjectID + 1 x Date + 1 x String + 1 x Int ~ **68 bytes per Document**
- size(**User**): 1 x ObjectID + 11 x String + 1 x Date + 1 x Int + N x size(Telemetry) = [пусть N = 50] ~ **3776 bytes per Document**
- size(**Class**): 2 x ObjectID + 4 x String + 1 x Int + N x String = [пусть N = 3] ~ **253 bytes per Document**
- size(**Comment**): 2 x ObjectID + 2 x String ~ **88 bytes per Document**
- size(**Record**): 2 x ObjectID + 2 x String + N x size(Comment) + 1 x Int = [пусть N = 2] ~ **268 bytes per Document**
- size(**Queue**): 3 x ObjectID + 7 x String + N x String + M x ObjectID + K x size(Record) + 2 x Boolean = [пусть N (количество групп) = 3, M (количество студентов) = 20, K (количество записей) = 40] ~ **11320 bytes per Document**
- size(**Discipline**) = 1 x ObjectID + 2 x String = **76 bytes per Document**

Посчитанные значения определяют **рост модели при увеличении объема каждого из документов**. Можно заметить, что наибольший размер имеют документы очередей Queue.

Определим размер БД в работе. Отметим, что в оценках выше присутствовали вложенные документы, которые не существуют как отдельные коллекции. Пусть в БД имеется:

- 1000 пользователей (User)
- 500 занятий (Class)
- 40 дисциплин (Discipline)
- 2000 очередей (Queue)
- На каждую очередь в среднем - 20 участников, 40 записей, на каждую запись - 2 комментария
- На каждого пользователя - 50 записей в телеметрии

1000 * 3776 + 500 * 253 + 40 * 76 + 2000 * 11320 = 26 545 540 bytes ~ 25 923 kB ~ 25.3 mB Тогда оценка размера БД: **25.3 мегабайт**.

Избыточность модели по сравнению с "чистыми" данными

Посчитаем объем памяти, который занимают дополнительные поля для каждого документа:

- Telemetry: 2 x ObjectID = **24 bytes per Document**
- User: 1 x ObjectID + N x (2 x ObjectID) = [пусть N (количество телеметрии) = 50] = **1212 bytes per Document**
- Class: 2 x ObjectID = **24 bytes per Document**
- Comment: 2 x ObjectID = **24 bytes per Document**
- Record: 2 x ObjectID + N x (2 x ObjectID) = [пусть N (количество комментариев) = 2] = **72 bytes per Document**
- Queue: 3 x ObjectID + M x ObjectID + K x (2 x ObjectID + N x (2 x ObjectID)) = [пусть M (количество студентов) = 20, K (количество записей) = 40] = **3156 bytes per Document**
- Discipline: 1 x ObjectID = **12 bytes per Document**

Оценим избыточность модели при том же заполнении БД, что в предыдущем пункте.

1000 * 1212 + 500 * 24 + 40 * 12 + 2000 * 3156 = 7 536 480 bytes = 7 359 kB = 7.1 mB

Определим процент избыточных данных: %изб = Nизб / Nобщ = 7536480 / 26545540 ~ 0.28

Таким образом, при стандартном использовалии базы данных в модели будет присутствовать около **28%** избыточных данных, включая идентификаторы и ссылки на документы. Такой высокий процент обусловлен большим количеством связей между документами и использованием идентификаторов для вложенных документов.

Формула объема для грубой оценки

Пусть V - объем БД в байтах, Nоч - количество очередей. Так как наибольший объем занимают именно очереди, то объем данных можно грубо оценить исходя из количества очередей. Также приличный объем уходит на

хранение данных о пользователе, поэтому примем допущение, что количество пользователей в 3 раза ниже, чем число очередей, чтобы выразить все через один параметр. В таком случае оценка объема БД:

V ~ 11320 * Nоч + (3776 * Nоч / 3)

V ~ 12578 * Nоч

Формула объема избыточных данных для грубой оценки

Пусть Vизб - объем избыточных данных в БД в байтах. Пользуясь соображениями из предыдущего пункта, получаем оценку объема избыточных данных:

Vизб ~ 3156 * Nоч + (1212 * Nоч / 3)

Vизб ~ 3560 * Nоч

Полученные формулы позволяют грубо оценить скорость роста модели и избыточности данных.

3.2. Структура реляционной СУБД

Схема SQL БД для ElQueue

Схема данных представлена на рис. 3.

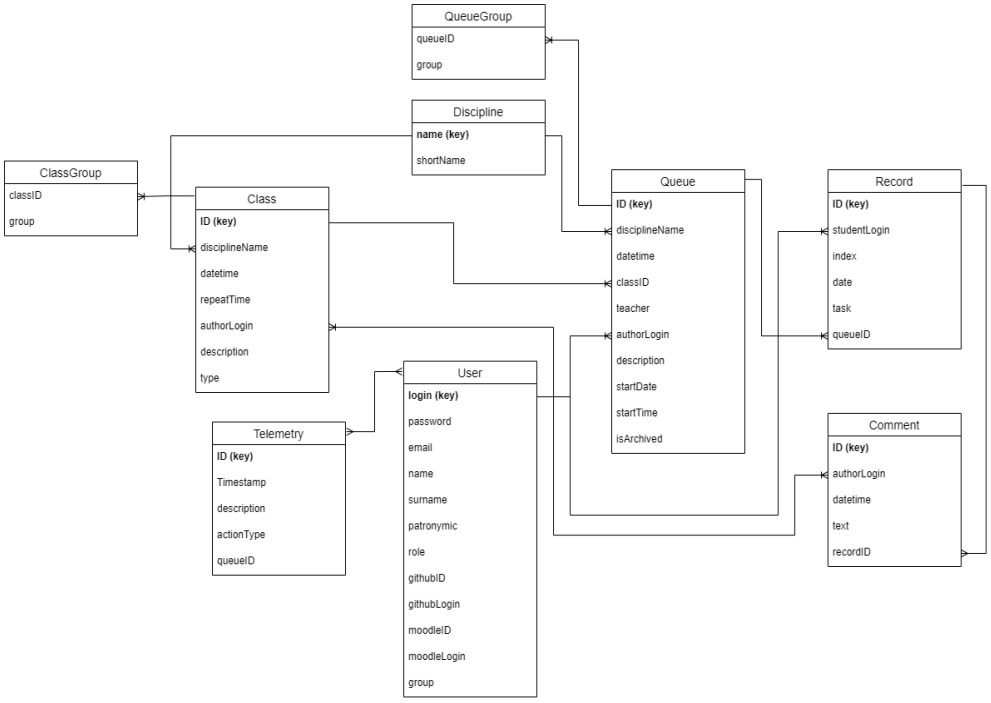

Рисунок 3 – Схема SQL DB

Список отношений

Описание сущностей (и оценка объема)

Условности

- Пусть строка в среднем имеет длину 16 символов и состоит из двухбайтных символов.
- Пусть в качестве идентификаторов используются BigInt (8 байт)
- В целом методика подсчета аналогична подсчету для НСУБД, поэтому подробности будут опущены

1. Сущность User

Сущность хранит основную информацию о пользователей (ФИО, группа (если есть), роль в системе, логин и т.д.)

- Примерный объем занимаемой памяти: 360 байт на строку
- Избыточный объем памяти: в явном виде отсутствует (0 байт)

2. Сущность Telemetry

Сущность хранит информацию о каком-либо действии пользователя (время, описание, типа действия, ID очереди, если действие происходило в ней)

- Примерный объем занимаемой памяти: 64 байт на строку
- Избыточный объем памяти (ID, queueID, userID): 24 байт на строку

3. Сущность Queue

Сущность хранит информацию об очереди

- Примерный объем занимаемой памяти: 170 байт на строку
- Избыточный объем памяти (ID, classID, authorLogin, disciplineName): 80 байт на строку

4. Сущность Record

Сущность хранит информацию о записи в очередь. Связь N-1 с сущностью Queue (Record хранит ID очереди)

- Примерный объем занимаемой памяти: 116 байт на строку
- Избыточный объем памяти (ID, studentLogin, queueID): 48 байт на строку

5. Сущность Comment

Сущность хранит информацию о комментарии к записи. Связь N-1 с сущностью Record (Comment хранит ID записи)

- Примерный объем занимаемой памяти: 112 байт на строку
- Избыточный объем памяти (ID, queueID, authorLogin): 48 байт на строку

6. Сущность QueueGroup

Сущность связывает очередь и номера групп. Связь N-1 с сущностью Queue (QueueGroup хранит ID очереди)

- Примерный объем памяти: 40 байт на строку
- Избыточный объем памяти (queueID): 8 байт на строку

7. Сущность Class

Сущность хранит информацию о занятии. Связь 1-N с сущностью Queue (Queue хранит ID занятия)

- Примерный объем памяти: 172 байт на строку
- Избыточный объем памяти (ID, disciplineName, authorLogin): 72 байт на строку

8. Сущность ClassGroup

Сущность связывает занятие и номера групп. Связь N-1 с сущностью Class (ClassGroup хранит ID занятия)

- Примерный объем памяти: 40 байт на строку
- Избыточный объем памяти (classID): 8 байт на строку

9. Сущность Discipline

Сущность хранит информацию о дисциплине (название, короткое название). Связь 1-N с сущностью Queue (Queue хранит название дисциплины). Связь 1-N с сущностью Class (Class хранит название дисциплины).

- Примерный объем памяти: 64 байт на строку
- Избыточный объем памяти: 0 байт на строку

Оценка удельного объема информации, хранимой в модели

Пусть в БД хранится следующая информация:

- 1000 пользователей (User)

- 500 занятий (Class)
- 40 дисциплин (Discipline)
- 2000 очередей (Queue)
- На каждую очередь в среднем - 20 участников, 40 записей, на каждую запись - 2 комментария
- На каждого пользователя - 50 записей в телеметрии

Тогда потребуется:

- 1000 записей в таблице User: 1000 * 360
- 50 * 1000 записей в таблице Telemetry: 50000 * 64
- 2000 записей в таблице Queue: 2000 * 170
- 40 * 2000 записей в таблице Record: 80000 * 116
- 2 * 40 * 2000 записей в таблице Comment: 160000 * 112
- 3 * 2000 записей в таблице QueueGroup (на каждую очередь в среднем - 3 группы): 6000 * 40
- 500 записей в таблице Class: 500 * 172
- 3 * 500 записей в таблице ClassGroup: 1500 * 40
- 40 записей в таблице Discipline: 40 * 64

Суммарная оценка объема памяти для выбранного заполнения БД: **31 488 560 bytes = 30 750 kB = 30 mB**

Суммарный избыточный объем: **12 976 000 bytes = 12 671 kB = 12.37 mB**

Определим процент избыточных данных: %изб = Nизб / Nобщ = 12 976 000 / 31 488 560 ~ 0.41

Таким образом, при стандартном использовалии реляционной базы данных в модели будет присутствовать около **41%** избыточных данных.

Формула объема для грубой оценки

Пусть V - объем БД в байтах, Nоч - количество очередей. Так как наибольший объем занимают именно очереди, то объем данных можно грубо оценить исходя из количества очередей. Также приличный объем уходит на хранение данных о пользователе, поэтому примем допущение, что количество

пользователей в 3 раза ниже, чем число очередей, чтобы выразить все через один параметр. В таком случае оценка объема БД:

V ~ Nоч / 3 * (360 + 50 * 64) + Nоч * (170 + 40 * 116 + 80 * 112 + 3 * 40) = Nоч / 3 * 3560 + Nоч * 13890

V ~ 15076 * Nоч

Формула объема избыточных данных для грубой оценки

Пусть Vизб - объем избыточных данных в БД в байтах.

Vизб ~ Nоч / 3 * (50 * 24) + Nоч * (80 + 40 * 48 + 80 * 48 + 3 * 8)

Vизб ~ 6264 * Nоч

3.3. Запросы к MongoDB и SQL БД

1. Получение данных об очереди, всех записей в очереди и всех комментариев к записям (по ID очереди)

Запрос актуален, так как при открытии страницы очереди необходимо, чтобы сервер вернул список записей и комментариев, которые будут отображаться в UI.

Пусть у очереди ID = 5.

MongoDB

Для получения информации необходим один запрос:

```
db.queue.find( { _id: 5 } );
```

Записи в очередь и комментарии к ним являются вложенными документами, поэтому входят в состав документа.

SQL

Для получения информации необходимо:

- Сделать запрос к таблице Queue

```
SELECT * FROM Queue
WHERE ID = 5
```

- Сделать запрос к таблице Record

```
SELECT * FROM Record
WHERE queueID = 5
```

- Сделать N (кол-во записей) запросов к таблице Comment

```
SELECT * FROM Comment
WHERE recordID = ?
```

Здесь ? - ID записи.

Всего: N + 2 запросов, где N - количество записей в очереди.

2. Получение списка очередей, в которые может записаться пользователь (студент) по номеру группы

Запрос актуален, так как при открытии страницы "очереди" пользователю представляется список очередей, в которые он может записаться, что определяется номерами групп, для которых предназначена очередь.

Пусть группа пользователя - 8382.

MongoDB

MongoDB имеет возможность поиска документа с массивом, в котором содержится определенный элемент: link

Для получения информации необходим один запрос:

```
db.queue.find( { groups: "8382" } );
```

SQL

Для получение информации также необходим один запрос, но необходимо использовать INNER JOIN.

```
SELECT Queue.*
FROM Queue INNER JOIN QueueGroup ON Queue.ID = QueueGroup.queueID
WHERE (((QueueGroup.group)="8382"));
```

3. Получение записей по ID пользователя

MongoDB позволяет искать по вложенным документам и делать фильтрацию вывода таким образом, чтобы возвращались только подходящие: link

Пусть ID пользователя - 5.

MongoDB

```
db.queue.find({ 'records.student.$iod': 5 }, {"_id": 0, 'records.$': 1});
```

SQL DB

В SQL БД также потребуется один запрос, так как есть таблица Record:

```
SELECT * FROM Record
WHERE studentID = 5
```

4. Получения списка ID преподавателей и количества созданных ими очередей

MongoDB

Запрос сложный, но все же один:

```
db.queue.aggregate([
   {$lookup:
     {
       from: "user",
       localField: "author.$oid",
       foreignField: "_id",
       as: "queues"
     }
   },
   {$unwind: "$queues"},
   {$group: {"_id":"$_id", count: {$sum:1}}}
]
)
```

SQL БД

Один запрос:

```
SELECT User.ID, Count(Queue.description) AS [Queue Count]
FROM [User] INNER JOIN Queue ON User.ID = Queue.author
GROUP BY User.ID;
```

5. Удаление пользователя по ID

Пусть ID пользователя = 5.

Mongo DB

Потребуется удаление очередей, записей, занятий, созданных пользователем

```
db.user.deleteOne( { _id : 5 } )
db.queue.deleteMany ( { "author.$oid" : 5 } )
db.class.deleteMany ( { "author.$oid" : 5 } )
db.queue.update(
  { },
  {
    $pull: { "records": { "student.$oid": 5 } }
  }
);
```

SQL DB

```
DELETE FROM User WHERE ID = 5
```

3.4. Сравнение нереляционной и реляционной базы данных

Сравнение удельных объемов при некотором заполнении БД

В табл. 1 приведено сравнение объемов всех данных в НСУБД и SQL-БД при заполнении равноценной информацией, а также сравнение избыточности данных.

Таблица 1

Параметр	MongoDB	SQL DB
Объем БД	26 545 540 bytes	31 488 560 bytes
Объем изб. данных	7 536 480 bytes	12 976 000 bytes
% изб. данных	28%	41%

В табл. 2 ниже приведено сравнение грубых оценок зависимости объема БД и объема избыточных данных от количества очередей Noч.

Таблица 2

Параметр	MongoDB	SQL DB
Множитель объема БД	12578	15076
Множитель объема изб. данных	3560	6264

В результате анализа можно сказать, что MongoDB лучше подходит для реализации электронной очереди, чем SQL база данных, так как использование вложенных документов позволяет сократить процент избыточных данных. Очередь - достаточно вложенная структура (очередь -> запись в очередь -> комментарий), поэтому именно объектно-ориентированные СУБД (MongoDB) позволяют наиболее компактно хранить информацию. Сравнение запросов представлено в табл. 3.

Таблица 3 – Сравнение запросов

Задача	Кол-во запросов в MongoDB	Кол-во запросов в SQL БД	Комментарий
По ID получить очередь, записи, комментарии	1	N + 2 (N - кол-во записей)	MongoDB значительно выигрывает
Получение очередей по номеру группы	1	1 (но с LEFT JOIN)	В MongoDB запрос проще и удобнее
Получение записей по ID пользователя	1	1	Равный результат
Получение ID пользователя и количества созданных им очередей	1 (сложный)	1	В SQL БД запрос несколько проще
Удаление пользователя	4	1	Благодаря каскадному удалению SQL БД справляется за меньшее число запросов

В результате анализа можно сказать, что MongoDB лучше подходит для базовых запросов. Для задач, связанных с очередями, записями, комментариями к ним, меньшее количество запросов создается в MongoDB по сравнению с SQL БД за счет использования вложенных документов. Однако SQL БД может оказаться удобнее и эффективнее, если необходимо находить статистику по пользователю (благодаря связям), а также она позволяет обеспечивать каскадное удаление (например, при удалении пользователей, удаляются созданные им записи, очереди, занятия).

3.5. Вывод

Анализ показал, что MongoDB хорошо подходит для реализации базы данных электронной очереди. СУБД имеет преимущество над SQL СУБД в занимаемом объеме данных, количестве запросов для базовых потребностей приложения. Тем не менее, SQL СУБД имеет ряд преимуществ, связанных с более нестандартными запросами (сбор статистики), а также с наличием каскадного удаления.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание

Веб-приложение электронной очереди ElQueue состоит из нескольких модулей:

- Авторизация

Авторизация пользователей происходит с помощью логина и пароля, также возможен вход в приложение с помощью аккаунта GitHub (OAuth). Модуль также предоставляет возможность регистрации пользователя. Страница входа представлена на рис. 4.

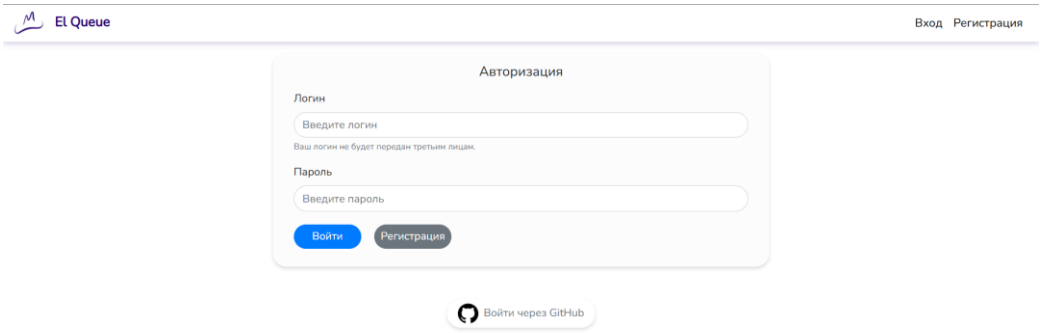

Рисунок 4 – Страница входа

Страница регистрации представлена на рис. 5.

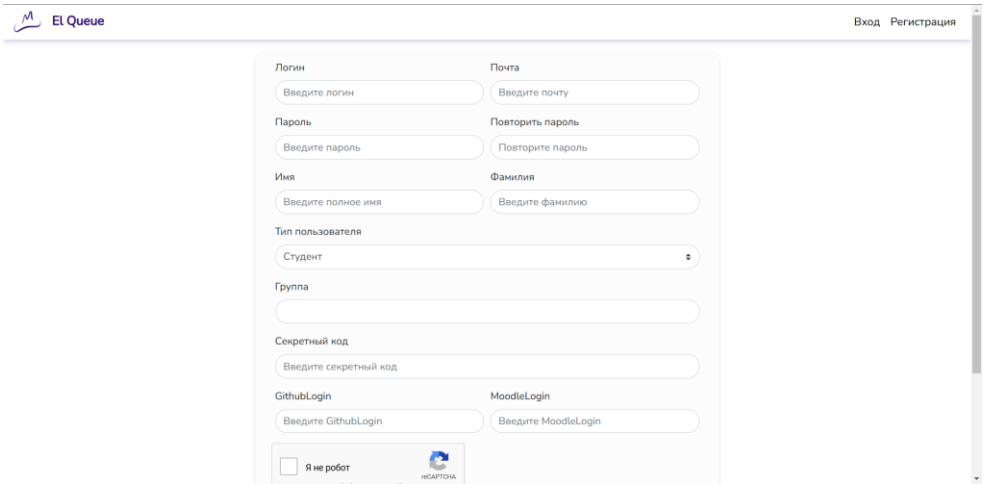

Рисунок 5 – Страница регистрации

- Очереди

Страница очередей представляет собой список карточек с возможностью фильтрации и действий с очередями: удаление (для авторов или администраторов), открытие. Страница представлена на рис. 6.

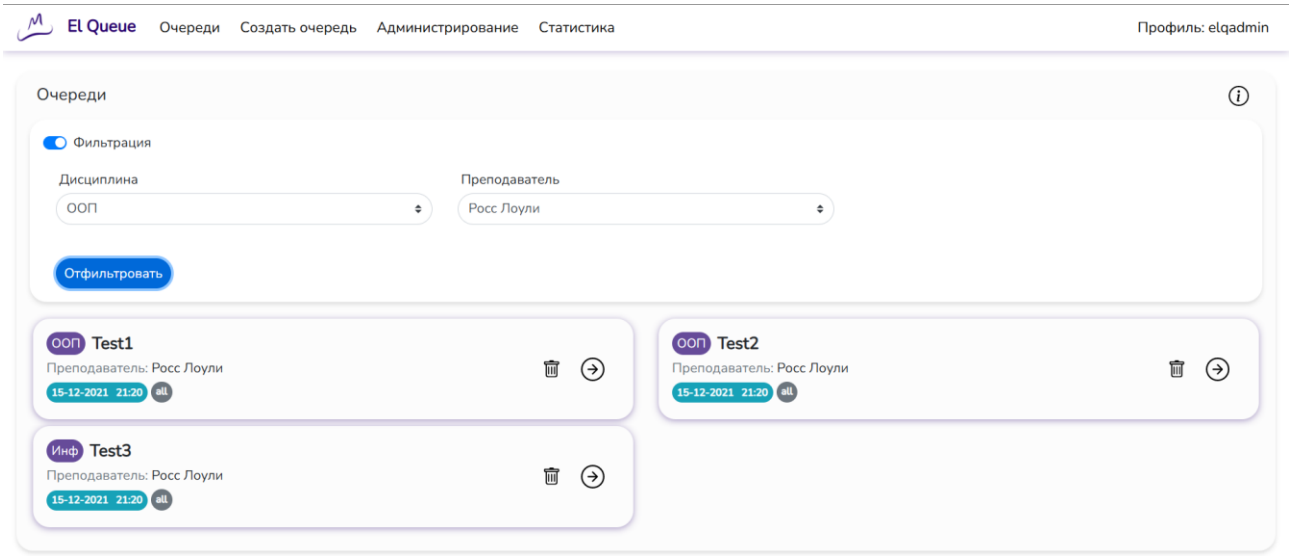

Рисунок 6 – Страница очередей

Страница очереди состоит из панелей информации об очереди, панели записи в очередь и упорядоченного списка записей в очередь. Возможен просмотр дополнительной информации о записи, комментирования, перемещения, удаления записи. Страница представлена на рис. 7.

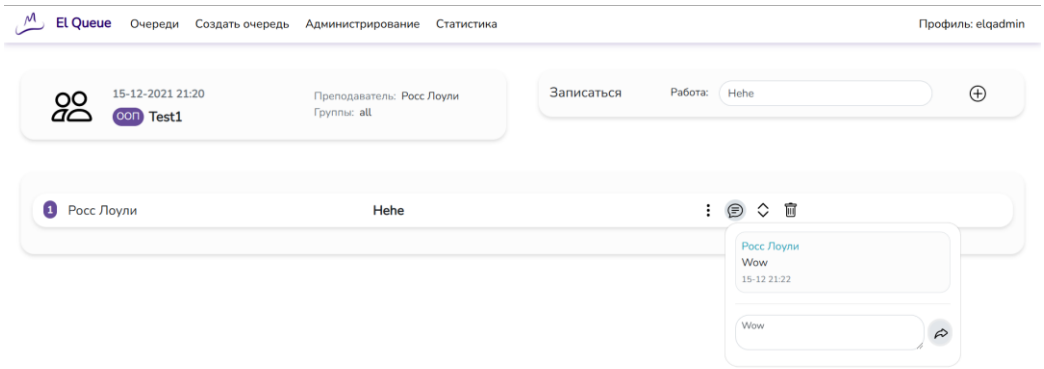

Рисунок 7 – Страница очереди

- Создать очередь

Страница создания очереди содержит поля ввода для новой очереди. Интерфейс представлен на рис. 8.

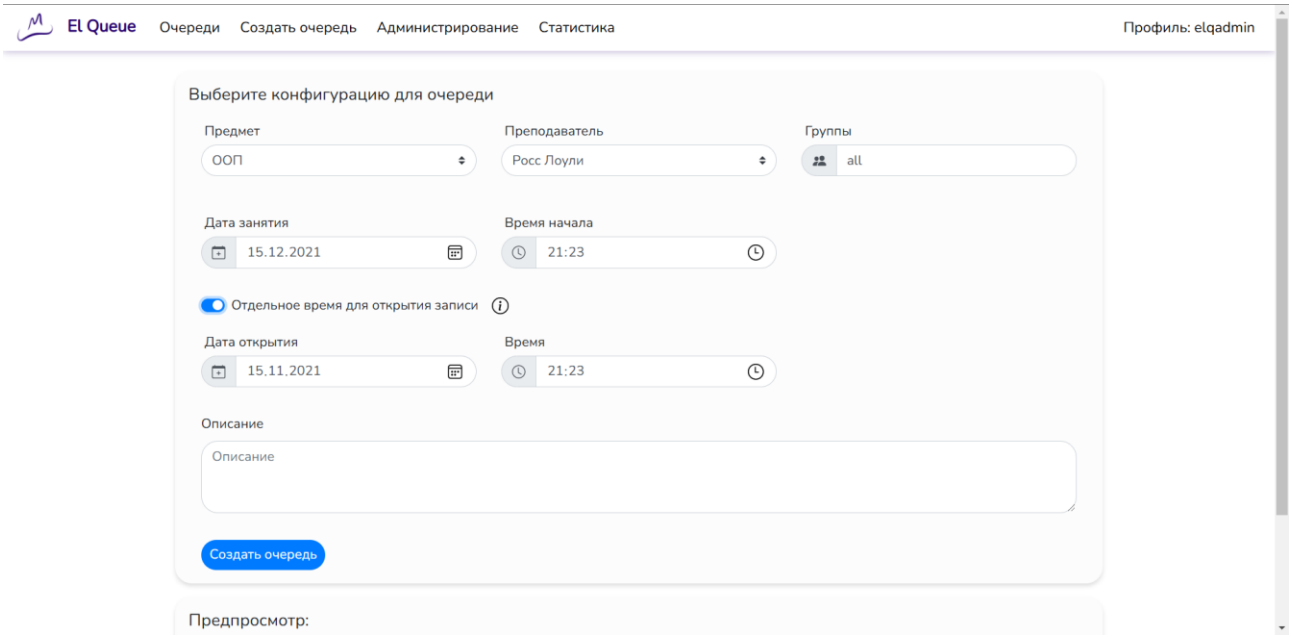

Рисунок 8 – Страница создания очереди

- Страница Администрирования

Страница состоит из карточек-ссылок на подстраницы: списка пользователей, занятий, дисциплин, секретных ключей. Также на странице возможен импорт базы данных из json файла и экспорт БД. Интерфейс представлен на рис. 9.

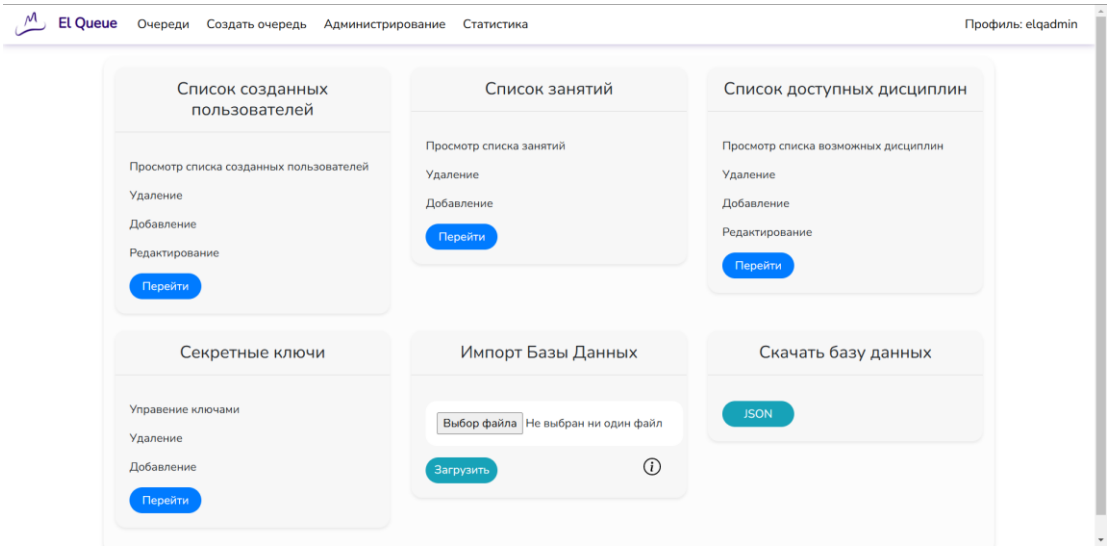

Рисунок 9 – Страница администрирования

Страница списка занятий содержит таблицу занятий, а также виджет расписания, в котором размещены занятия согласно правилам их повтора. Интерфейс представлен на рис. 10.

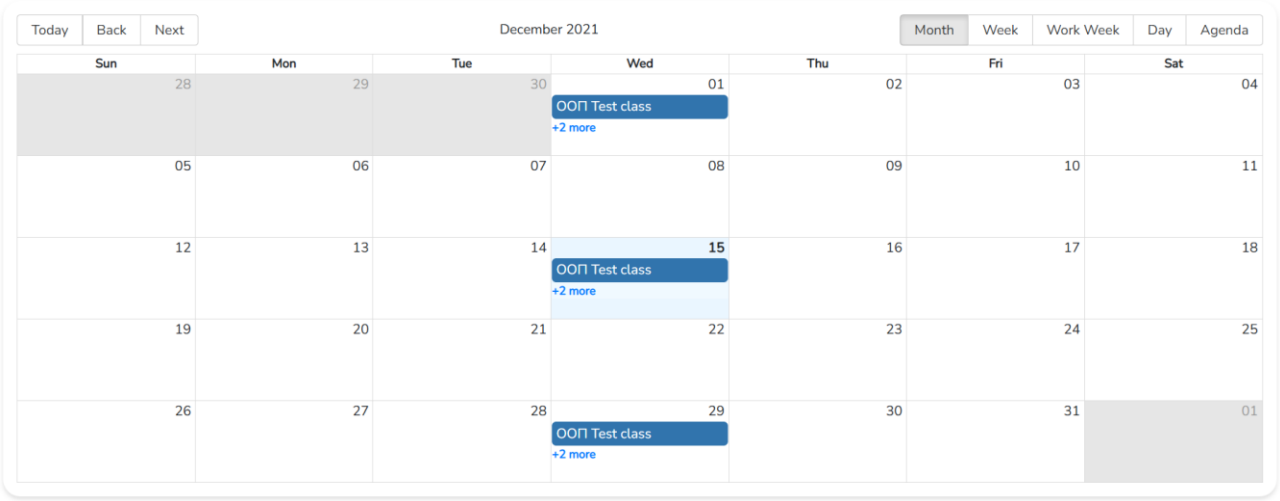

Рисунок 10 – Страница списка занятий (фрагмент)

- Статистика

Страница статистики позволяет получать статистику по очередям по фильтру. Интерфейс представлен на рис. 11.

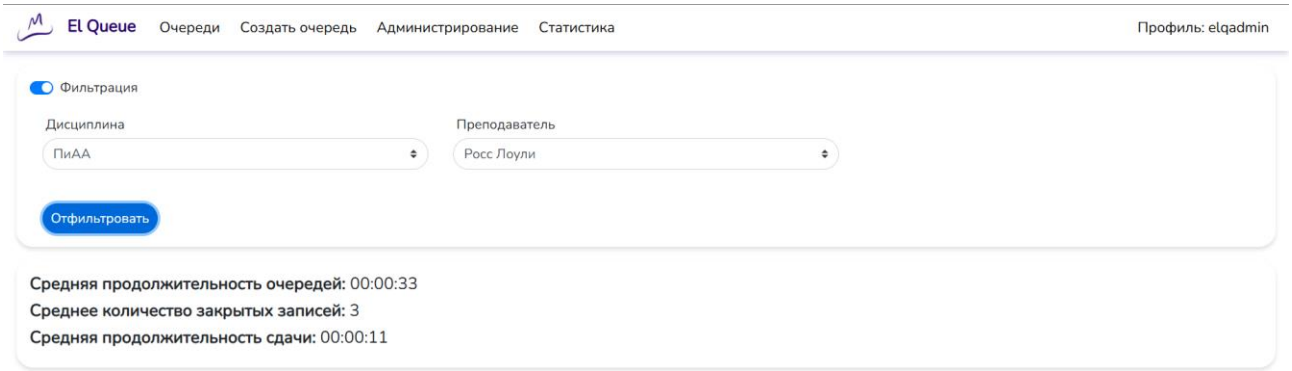

Рисунок 11 – Страница статистики

4.2. Использованные технологии

Использованная СУБД: MongoDB

Фреймворк для Backend: Flask (Python)

Фреймворк для Frontend: React

Контейнеризация: Docker

Дополнительные библиотеки: Bootstrap, Flask-JWT, mongoengine, Socket-IO, Flask-Dance.

5. ВЫВОДЫ

В результате выполнения курсового проекта было разработано приложение для организации электронных очередей с использованием СУБД MongoDB, выполняющее основные задачи:

- Предоставление преподавателям интерфейса для организации электронные очереди для вышеуказанных учебных процессов и контроля проведения занятия по очереди, планирования систематического проведения очередей.
- Предоставление студентам интерфейса для просмотра доступных очередей, записи в очередь, комментирования очереди, просмотра статистики по очередям определенного типа.
- Предоставление администраторам интерфейса для контроля пользователей, просмотра статистики действий пользователя, архива записей, бэкапа и восстановления базы данных.

Дальнейшие шаги для улучшения решения:

- Внедрение Socket-IO в большее количество страниц приложения для динамического обновления данных
- Хранение дат в специальном формате в БД (изначально был выбран формат строки, не являющийся удобным для сортировки)
- Продвинутые поиск и фильтрация по телеметрии пользователей
- Более детализированная статистика и аналитика

6. ПРИЛОЖЕНИЯ

6.1. Сборка и развертывание приложения

- Сборка Frontend
  - Необходима установленная NodeJS
  - В папке /app/reactApp запустить команду npm install
  - Запустить команду npm start
  - Перейти по адресу localhost:3000
- Сборка Backend
  - Необходим установленный Python 3.9
  - Запустить скрипт app/back/scripts/setup.bat
  - Для последующих запусков использовать скрипт ./run.bat
- Развертывание приложения
  - Необходим установленный Docker, Docker-Compose
  - В папке /app запустить команду docker-compose build –no-cache
  - Для переустановки запустить скрипт ./recreate_docker_constainers.sh